NASA Contractor Report 181905

Flight Crew Aiding for Recovery From Subsystem Failures

E. Hudlicka, K. Corker, R. Schudy, and S. Baron

BBN Systems and Technology Corporation
Cambridge, MA 02138

Contract NAS1-17335

January 1990

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

NASA Contractor Report 181905

Flight Crew Aiding for Recovery From Subsystem Failures

E. Hudlicka, K. Corker, R. Schudy, and S. Baron

BBN Systems and Technology Corporation
Cambridge, MA 02138

Contract NAS1-17335

January 1990

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

Table of Contents

List of Figures

Abstract

This document discusses some of the conceptual issues associated with pilot aiding systems and describes an implementation of one component of such an aiding system.

The first part of this document focuses on aiding systems design issues. An aiding system, by definition, acts to assist the crew. It is therefore essential that the format and content of the information it presents to the crew be compatible with the crew's mental models of the task. The basic thesis we propose is therefore that in order to cooperate effectively, both the aiding system and the flight crew should have consistent information processing models, especially at the point of interface. We have selected a general information processing strategy, developed by Rasmussen, to serve as the bridge between the human and aiding system's information processes

The second part of the document describes the development and implementation of a model-based situation assessment and response generation system for commercial transport aircraft. The current implementation is a prototype which concentrates on engine and control surface failure situations and consequent flight emergencies. The aiding system, termed RECORS (Recovery Recommendation System), uses a causal model of the relevant subset of the flight domain to simulate the effects of these failures and to generate appropriate responses, given the current aircraft state and the constraints of the current flight phase. Since detailed information about the aircraft state may not always be available, the model represents the domain at varying levels of abstraction and uses the less detailed abstraction levels to make inferences when exact information is not available. The structure of this model is described in detail, followed by a description of the types of reasoning supported by the knowledge contained in the model.

RECORS is intended to be integrated with the FaultFinder system currently under development at NASA-Langley.

1. Introduction

This is the final report in the Intelligent Aids for Flight Crew Tasks project (Contract No. NAS1-17335). In previous work [Baron and Feehrer, 1985], we analyzed flight crew tasks, surveyed the state of AI research, examined human factors issues relevant to intelligent aiding systems and aids, and identified several research needs in AI. That effort suggested the desirability and importance of intelligent interfaces that would be capable of managing the information presented to the crew in efficient and effective ways to facilitate communication between the crew and other on-board intelligent systems. Such intelligent interfaces, and the necessary underlying problem-solving capabilities, have been the subject of our subsequent efforts described in the interim report for this project [Hudlicka, et al., 1987]. This report is divided into two principal sections. First, we discuss the definition and design requirements of an intelligent aiding system. In particular, we describe a model for crew information processing activities that is instrumental in guiding the development of both the intelligent interfaces and the underlying information processing by the intelligent aiding system. The second part of this report focuses on the architecture and functionality of a prototype of such an underlying aiding system, the Recovery Recommendation System, or RECORS. Although our approach is general in nature, our efforts thus far have focused on situations involving engine and control surface failures in commercial air transport operations. The examples and test cases used are therefore drawn from that domain.

1.1 Background

In the course of any flight, commercial air-transport pilots engage in a myriad of perceptual, information processing, and behavioral tasks. From procedurally-defined control and communication tasks, to judgments and decision making for adaptation to changing craft and environmental conditions, to complex perceptual-motor behaviors in the control and guidance of the aircraft, the tasks of the flight crew are dynamic, adaptive, and complex. In emergency situations, the flight crew must typically perform such tasks in unfamiliar situations, and under

time pressure. Pilot aiding systems for these critical situations should place minimum demands on the pilot and should anticipate the flight crew's needs in order to provide intelligent cooperation. In less urgent but novel situations, the flight crew would also benefit from specialized skills which they may not have, but which could be embodied in expert aiding systems.

In early aviation, pilots had a nearly complete understanding of the aircraft and its systems, and could, as was often necessary, diagnose failures in the aircraft systems and repair, or work around, those failures. The systems in turn had no model of, or ability to adapt to, the situation or the pilot. In time this has changed. Current aircraft systems perform many complex functions and adapt in many ways to the situation and to the explicit commands of the flight crew. Simple system self-knowledge, in the form of built-in-test and system diagnostics have been implemented to monitor many of the more complex system functions [Wiener, 1988].

The work described in this report has as its goal further development in pilot aiding systems. To this end, the model embedded in the RECORS system includes knowledge of aircraft systems, effectors, the physical forces acting on the aircraft, the flight profile characteristics, and the specific goals associated with the different flight phases. This knowledge is used both to simulate the effects of faults in the aircraft systems or effectors, and to generate responses to emergencies. Further work in this area will also include knowledge of the flight crew's models of the aircraft state in order to tailor the interface information to avoid overwhelming the pilot. The development, then, is toward more detailed knowledge of the systems being monitored, knowledge of the interactions (physical, functional, temporal) among those systems, knowledge of the flight situation in which the monitoring is being conducted, and knowledge of the condition of the flight crew that is being assisted.

In order to motivate our discussion and provide a factual basis for system development, we refer throughout the report to incident/accident examples drawn from National Transportation Safety Board (NTSB) reports, and to pilot interviews. Descriptions of these accidents are in Appendix A.

1.2 Objectives of Research

In our view, the principal objectives of intelligent pilot aiding systems are:

- improving pilot situational awareness
- reducing the time and pilot workload required to identify correct behaviors for a given situation;
- reducing the likelihood of error in the pilot's identification and execution of those behaviors and,
- assisting the pilot in executing those behaviors.

An additional aim of aiding systems is to embody the expertise of many highly skilled specialists in order to assist flight crews in situations that are outside their domain of experience and expertise. An example of the utility of such on-board expertise can be found in Appendix A, Incident 1.

1.3 Overview of Approach

Our approach to meeting the objectives outlined above involved the following major activities:

- Knowledge acquisition: Identifying the situations where pilots could use help, and the tasks with which they could use help;
- Requirements analyses: Deriving knowledge and functional requirements for intelligent aiding system and defining conceptual information-processing framework for cooperation between the flight crew and the intelligent aiding systems.
- Prototype Implementation: Designing and implementing a prototype intelligent system capable of providing assistance to pilots during emergencies. The system is intended to function in conjunction with the FaultFinder diagnostic system currently under development at NASA-Langley [Abbott 1988].

Knowledge Acquisition. In order to focus our acquisition of the domain knowledge, we have examined the National Transportation Safety Board (NTSB) reports involving engine failure on commercial transport aircraft. These reports are summarized in Appendix A. The conditions of engine failure and the flight crew response (judged appropriate or in error in the

NTSB report) were identified. Incidents and accidents were characterized according to failure type (components, single or multiple failures, primary and ancillary damage as a function of engine loss) and according to flight phase (rollout, takeoff, climb, cruise, descent, or landing). In addition to these reports we have conducted several interviews with airline pilots and have consulted flight training manuals. From these sources we have developed an initial knowledge base describing the aircraft subsystems and effectors and their relationships to the flight profile characteristics. This knowledge base also includes specific undesirable situations and the associated prescribed responses.

Due to the breadth of the overall task of pilot aiding systems, both the knowledge acquisition and the system design and implementation are necessarily iterative tasks. In the future, we hope to expand the knowledge base to include information from other experts who have the knowledge and skill required to deal with unusual situations, such as engineering test pilots, aeronautical engineers, and experts in aircraft systems such as avionics, engines, and flight controls.

Requirements Analyses. Based on initial knowledge acquisition studies, we developed a unifying information processing framework. The information processing framework serves two essential roles. First, it provides a *conceptual* foundation for the definition and implementation of intelligent flight crew aiding systems. Second, it provides a *computational* foundation for the reasoning which the intelligent aiding system can perform about the flight crew needs, about the situation, and about the dialogs and tasks shared by the flight crew and aiding system. The framework is described in the context of a *human* information processing model in Chapter 2, and in terms of *computer* information processing models in Chapters 3 and 4.

The human information processing model we are proposing is focused on pilot behavior initiated by the pilot's assessment of the situation. There are five basic premises in our model of the situation-response component of pilot behavior.

1. Complexity, time constraints, and the high cost of error require that airline pilots respond to most situations following carefully designed procedures.
2. Pilots assess these situations in terms of a moderate number of situation types.
3. Procedures are associated through training with the situation types for which those procedures are appropriate.

4. Pilots typically recognize any given situation in terms of several different situation types.
5. Pilot use priority and other techniques to resolve the conflicts between the different procedures associated with the different situation types which together characterize the situation.

The computational models described in this report include both this fast processing associated with skill-based behavior, termed stimulus-response processing, and information processing requiring deeper knowledge of the domain and utilizing more of the pilot's resources, termed knowledge-based processing [Rasmussen, 1984]. The computational model implementing the stimulus-response type of processing is described in Chapter 3. The knowledge-based processing is implemented in the RECORS system and is described in Chapter 4.

Prototype Architecture. The design of the aiding system prototype was driven by the following considerations:

- That it integrate with the FaultFinder diagnostic system currently under development at NASA-Langley.
- That it be capable of making general inferences when detailed information about the state of the aircraft is not available.
- That it be able to reason about novel and multiple faults.
- That it be able to derive appropriate responses to situations resulting from such faults using the underlying principles of the domain.

These requirements led to a model-based system architecture which is described in detail in Chapter 4. Briefly, this architecture is based on a causal model of the flight domain. This model represents the relevant aspects of the domain (aircraft structure, subsystem behavior, flight characteristics) as objects in a frame-based representation. The causal relationships among these objects are stored in the object's slots.

These causal relationships are expressed at two levels of abstraction: binary level, which indicates whether the objects are causally related or not, and a qualitative level, which represents the nature of this causal relationship in terms of "directly-proportional" or "inversely-proportional". Having two levels of abstraction available for expressing causal relationships allows the model to support inferencing with input data which varies in the amount of detail.

The information in the model is expressed in a format which supports different types of inferencing. The causal relationships are represented as constraint expressions which can support value propagation in either direction. Propagating values forward through these expressions allows the simulation of faults on the aircraft effectors and flight characteristics. Propagating values backward through these expressions supports the derivation of possible actions to achieve a specified goal (typically a desired flight characteristic). The format of the knowledge is also suitable for explanation, which is particularly important when a system plays the role of an assistant to a human.

Chapter 4 describes in detail both the structure of the model, the rationale behind the structure, the details of the qualitative reasoning, and the types of inferencing the model supports to aid the flight crew in both situation assessment and response derivation.

2. A Cognitive Model of Situation-Response Behavior

In this chapter the pilot information processing model is first described in a general way from the perspectives of human factors and cognitive science. In Chapter 3 we describe in computational terms the situation-response portion of this model using techniques from artificial intelligence. Chapter 4 describes the design and implementation of the deep (knowledge-based)[1] processing portion of this model in the context of a prototype aiding system which simulates the effect of a failed effector (engine, control surface) on the flight profile and suggests an appropriate response.

2.1 Introduction

Aircraft flight crews engage in complex cognitive, perceptual, and psychological tasks throughout most phases of flight. This task environment is made even more complex by unanticipated anomalies or system failures to which the crew must adapt. The addition of time-critical situation assessment and decision loads in the face of emergency conditions during critical phases of flight stresses the human capacity to meet the system's demands. We have developed a model of the human information processing functions as they are applied to meeting the demands (on the flight-crew) of air transport operations.

Human information processing has been characterized as occurring at various levels, and in differentiable stages. The basis of processing is selected in response to the time and resource constraints imposed by the environment and by the level of expertise the human brings to this particular processing task. In keeping with the automated nature of air transport systems, our characterization of flight-crew information processing derives from studies of humans interacting with large-scale, semi-automated, dynamic systems in which the role of the human is predominantly that of a monitor or supervisor of sensor and control systems [Rasmussen, 1984; Woods, 1986]. In such systems, information processing can be considered to be based on:

[1]The term "knowledge-based" in this chapter is used in the sense described by Rasmussen [1983] to mean processing which is not done automatically but requires some reasoning. In the AI literature this type of processing is referred to as "deep reasoning" or "reasoning from first principles" [Davis, 1982, Michie, 1981-1982].

- "knowledge" in which operator models of the system process, judgment and decision making contribute to the identification and accomplishment of an operator's goals,
- "rules" by which the characteristics of a situation are identified as belonging to a set of stored "situations" for which actions and responses are known, but for which procedures need to be tailored to the specific attributes of the situation, and
- "skills" in which limited packets, or sets of behavior are applied to specific stimuli in the environmental situation, with little or no reasoning effort applied to their generation or modification.

We see these levels as occurring in a natural progression with fairly fuzzy boundaries between them. Intelligent aiding should take place across levels.

We will first describe the a class of information processing, which we term *situation-response* behavior. The scope of the situation-response model is *rule-based* behavior, that is, pilot behavior for which the correspondence between situations and applicable procedures has been established by engineering and training.[2] The focus of our research on situation-response behavior is motivated by evidence that by human pilots in time-critical situations can contribute to accidents [Rasmussen 1986]. This evidence comes from accident analyses which suggest that abstract reasoning may shift attention from flight-critical tasks, and that deep reasoning under stress, from necessarily incomplete information and incomplete abstract models, can produce results which are significantly and sometimes fatally inferior to those derivable from engineering studies, experience, and experiment. In order for pilots to accomplish their difficult tasks without making mistakes in practice, it is desirable that they follow procedures which have been carefully engineered, thoroughly trained, and frequently practiced. These engineering, training and practice requirements limit the number and flexibility of the procedures which pilots have available. Triggering situations can be expected to fall into classes which derive much of their structure from the set of response procedures.

Situation-response behavior is a type of behavior in which there is a rapid assignment of a response schema to a set of stimuli that have been assembled (through training) into a trigger for the response. Figure 2-1 illustrates an overall diagram of this type of processing. Situation-

[2]In the prototype implementation we have represented this type of information processing in a causal model in order to increase the class of situations that can be analyzed.

response behaviors are assembled and stored for rapid access and activation without requiring deep or novel reasoning. The links between situation characterization and response initiation are established by skill development processes such as planning, rehearsal, evaluation, trial and error, training and practice. One advantage of situation-response behavior is the efficiency, in terms of time and cognitive resources expended, with which some (hopefully appropriate) behavior can be initiated. The disadvantages of such behavior lie in the potential for inappropriate situation classification, the inability to handle novel or multiple faults, and in the cost for development and storage of a sufficiently large set of situations and response procedures to adequately deal with a complex and performance-critical task environment.

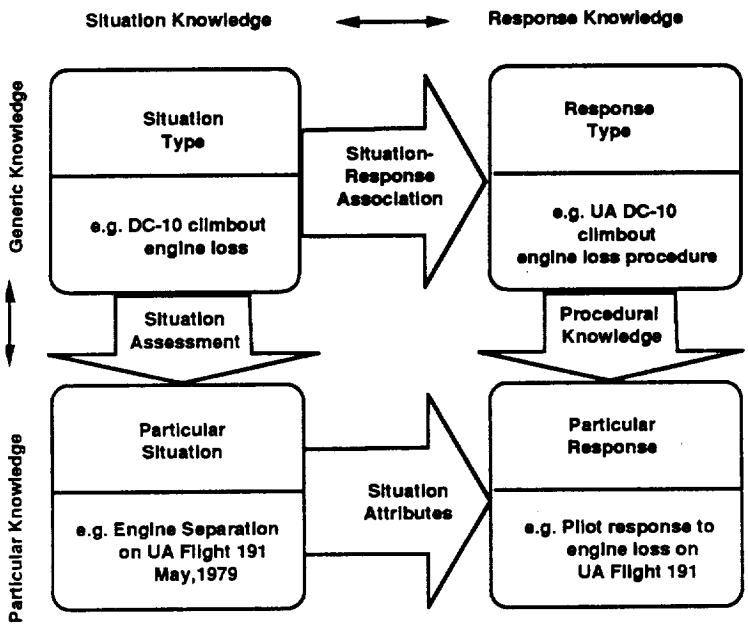


Figure 2-1: The Situation-Response Information Processing Model

2.2 General Model

Figure 2-2 schematically describes the overall processing steps for a real-time intelligent agent such as a pilot or pilot aiding system. The vertical, abstraction, axis in the figure corresponds to the extent to which the processing is removed from the concrete sensors and effectors. Movement along the horizontal axis represents progression of the processing from the sensors to the effectors. Knowledge-based information processing is at the top of this figure, rule-based processing in the middle, and skill-based processing along the bottom. The ovals in the figure represent states of knowledge, and the labeled arcs between those states represent processing which implements the indicated transitions between the states of knowledge. Different knowledge representation requirements are associated with the different states of knowledge, and different computational requirements are associated with the different classes of transitions. The particular states of knowledge and processing steps should not be interpreted as required functional partitions for particular systems. The actual flow of processing in nontrivial systems involves many more stages than are shown. Also, many more pathways may exist between the indicated states of knowledge, and other states of knowledge may be important for some applications; these other pathways are summarized by the three classes of shortcuts listed within the three large arrows.

2.3 Processing Without Shortcuts

The discussion below follows the information flow in Figure 2-2 from sensors to effectors, taking no shortcuts, and alternating between states of knowledge and the processing steps which accomplish the transformations between those states.

Sensors. The sensor state of knowledge is just the sensor data and other inputs to the intelligent agent. This input data includes "self" information from sensors such as control sensors (e.g. actuator positions, velocities), system status data (e.g. self-test and built-in-test results), and internal environmental sensor data (e.g. temperatures, pressures, accelerations, flow rates, fuel quantity, vibration, and power supply voltages). This input data also includes

environmental information derived both from sensors associated with the agent and from other agents. This input state knowledge may also include considerable information from any other agents with which this agent is closely cooperating. In commercial aviation this sensor information includes engine sensors (N1, N2, EGT, etc.), sensors for aircraft subsystems, environmental sensors, data from ground-based systems, and information from other pilots (PIREPS) as well as intra-cockpit communications.

Perception. Perception is the mapping of the sensor data into symbolic models of the entities and states of relevance to the intelligent agent. This signal-to-symbol processing includes diagnostic processing up to the level of system status. This is the process whereby the pilot scans his instruments, scans out of the cockpit, listens to radio transmissions, and assimilates this information into his "internal representation' of the state of the aircraft and the state of the airspace.

Entity and State Descriptions. The output of perceptual processing is a symbolic description of the state of the agent, the environment, and plans or cooperating agents. In current knowledge representation technology these states could be represented in terms of instances of archetypical entities and states, with their attributes refined from the current sensor data, or derived from the input data. This is essentially the instantiation of the pilot's general model of his aircraft.

Assessment. Assessment includes the processing stages in which the significance of the current situation is derived, in terms of the impact of the situation on the agent, its goals, plans, and actions. Assessment processing proceeds through the generation of successively more abstract descriptions of the situation attributes and of the situation as a whole.

Situation Descriptions. Situation descriptions are a proper superset of entity and state descriptions. Because of the scope of situation descriptions, it may no longer be feasible to describe situations in terms of single archetypical situation types. Thus situation descriptions may take the form of interrelated instances of different situation types, each of which describes some aspect of the total situation. For example, a normal take off situation may be augmented by a strong cross wind situation to describe a weather modified aircraft configuration.

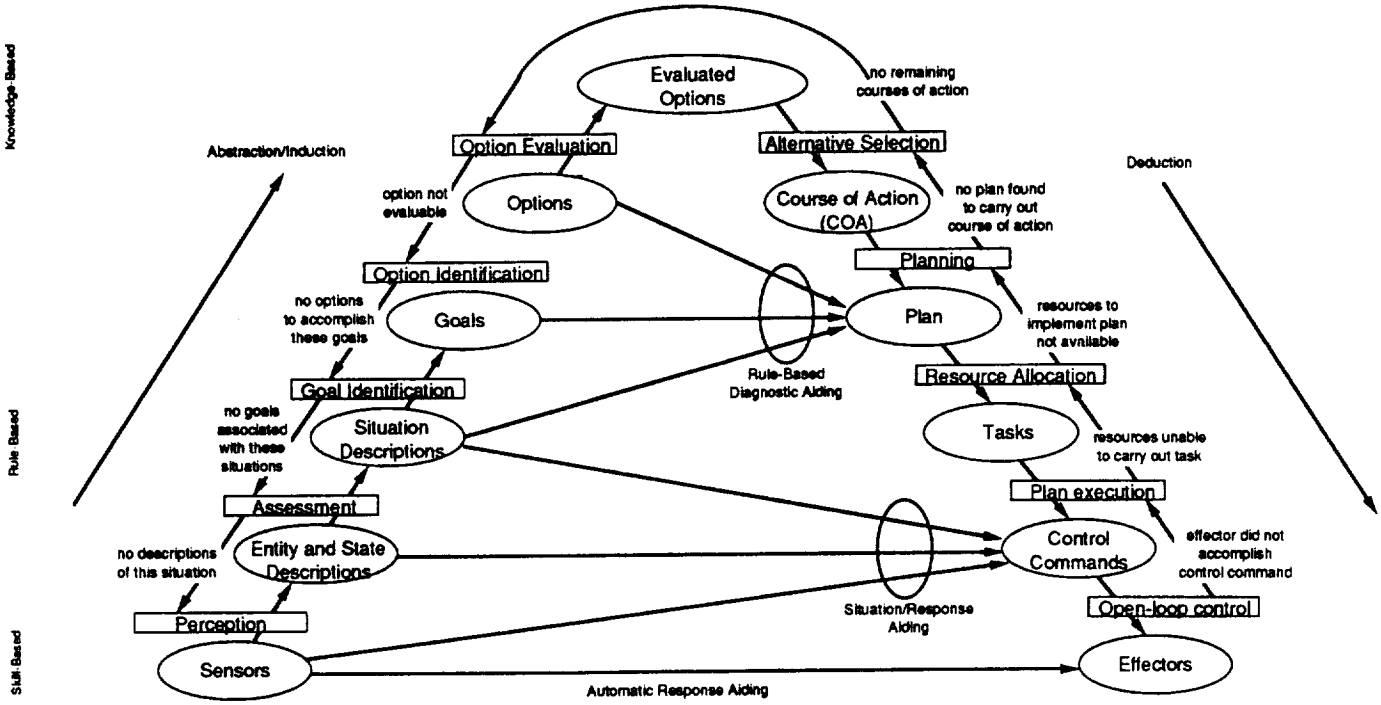


Figure 2-2: General Information Processing Model with Shortcuts

Goal Identification. Goals may enter at any level before this goal identification processing stage, but they must be defined by the end of this stage, so that the goals can be used to drive the option identification step. Goals direct and focus behavior. One way for behavior to adapt to the situation is for the goals to depend on the situation. While very general static goals such as survival may be useful in strategic planning, more specific and more near-term situation-dependent goals are more effective in driving behavior. Goal identification proceeds from assessments of the situation and from more global goals contained in the plan structure. Since different aspects of the situation may dictate different goals, resolution of goal conflicts may be required.

Goals. Goals link descriptions of future states, situations, and actions with statements of the desirability of those states, situations and actions. Since goals involve not only states, but also situations, actions, and the temporal and other relationships between states and situations, the representation for goals shares many of the characteristics of plans.

Option Identification. With descriptions of the state, situation, and goals, feasible behavioral options can now be generated.

Options. Options are descriptions of potential courses of action for the intelligent agent. Option descriptions may include state and entity descriptions, situation descriptions, and goals. The main additions are the models of time and decisions necessary to support the representation of courses of action.

Option Evaluation. The option evaluation step is the detailed application of the goal criteria against the identified options. Generally all that is desired is some optimal set of courses of action. The technologies of heuristic search (using admissible heuristic functions), optimization theory, and operations research can each address option evaluation problems in different domains, but no efficient general methods for option evaluation are currently known.

Evaluated Options. Unless powerful abstractions, admissible heuristics, and other means are available for abstracting and pruning the space of options, only a small set of the whole family of options can be evaluated. In order that the behavior be optimal, all attributes of evaluative interest must be captured in the representation of the options. Thus subsequent representations may be restrictions of this representation of options.

Alternative Selection. The alternative selection process may be goal-driven, if the goals impose a total ordering on the options, or it may involve the invocation of additional criteria to select amongst an equivalence class of best options. One computer implementation of the alternative selection process is as a pruning of a semilattice of evaluated options. The alternative selection process retains the root (representing the current state), and prunes all actions which cannot be part of an overall optimal course of action.

Course of Action. The course of action is a subset of the evaluated options. If the agent is operating in an uncertain environment involving unpredictable agents, or other sources of uncertainty, then the course of action may include branches to accommodate the generically different responses during plan execution. Thus the course of action, like the options, evaluated options, plans, and tasks, may be represented as a semilattice, with a root representing the current state, and nodes representing branches and joins of situations during possible plan executions.

Planning. The planning process is the expansion of the course of action to the level of operations to be performed. Note that many of the computational steps which would be considered as steps in non-real-time planning, such as option identification and evaluation, may have already been performed to support option evaluation and the selection of the course of action.

Plan. The plan representation may include entity and state representations, situation representations, decision criteria, and alternative options. From a representational standpoint it is thus equivalent to the evaluated options representation. The plan differs from that representation in two ways. First, it is uniformly expanded to the level necessary to support resource allocation processing; it thus must include all parallel operations, and detailed models of timing. Secondly, the plan includes only those options which passed the alternative selection process. The plan is thus more detailed than the course of action, but includes fewer options than the evaluated options.

Resource Allocation. Resource allocation binds specific resources to specific actions in the plan. Some resource conflicts may have been resolved earlier in the processing to evaluate the course of action and to develop the plan; the resource allocation process completes those

allocations. Resource allocation processing completes the development of task descriptions to support plan execution.

Tasks. The task representation is the plan representation expanded to the level of specific tasks for specific resources. Task descriptions include any parameters, other than inputs, required by the execution procedures. These task descriptions include sensing tasks, decision tasks, information processing tasks. It may also include arming and conditioning of real-time shortcuts.

Task Execution. Task execution follows the procedure for tasks of that type. This involves task initiation, performance monitoring, and parameter adjustment. Task execution can be influenced by the results from decision tasks, and by feedback from other running tasks.

Control Commands. The output of the task execution processing are control commands to the effectors and other resources. These control commands include all information necessary to determine the operation of those resources.

Control. Control processing may involve servo-loops, or can operate with limited or no feedback.

Effectors. Effectors include all resources under control of the agent, including information processing, actuators, and sensors.

It will generally be the case that the processes and representation described above are incomplete and ambiguous in some ways. For example, sensor data may be incomplete or missing or situations may not be unambiguously defined. Feedback and iteration from later stages of processing to earlier ones may serve to refine situations, goals, etc. In addition, efficiency can be improved by implementing feedback from later processing stages to control processing at earlier stages. This cascading can take the form of the integration of later functions into earlier functions, where it can help narrow the number of options considered, or it can take the form of specific information fed back from later stages when those stages have examined partial results from those earlier stages. An example of the processing of the first sort is the integration of resource allocation constraints into the option generation and evaluation stages.

An example of the second sort is feedback from the alternative selection function about partially evaluated options.

2.4 System Aiding in Response Processing

Evidence suggest that humans will move from various levels of analysis directly to various levels of response via so called processing shortcuts (see Figure 2-2 and Figure 2-3. Situation-response shortcuts, which are the focus of our model, move rightward from the vicinity of the *situation description* level of analysis to the vicinity of the *tasks* level of execution. For example, the behavior of a skilled transport pilot during takeoff may be determined almost entirely by his perception of the situation as a standard takeoff from that airport and by his procedures for taking off from that airport. Implementing behavior differing from established takeoff procedures depends on the pilot's recognizing that the situation is no longer solely, or most appropriately, described as a standard takeoff situation and using his knowledge of alternative situations and procedures to respond appropriately to this new situation.

The example below, taken from the well known DC-10 crash at O'Hare Airport in May 1979, serves to illustrate the role of situation recognition. Before engine separation the situation was described in terms of the normal takeoff situation types. At engine separation the engine-loss-climbout situation type also correctly described the situation. The engine-loss-climbout procedures require flight crew attention to airspeed, bearing, climb-rate, thrust-compensation, and behaviors designed to compensate for the engine loss and bring the aircraft to a safe altitude and flight path. Unfortunately, the situation type of engine-loss-climbout did not fully describe the situation. Retraction of left wing outboard slats placed the flight in a critical stall-regime situation. In low-speed flight any such flight control problem is an emergency of the highest priority, requiring immediate action. The procedures for such low speed flight control emergencies are directed toward increasing air speed in order to increase control effectiveness, stall margin, and maneuverability. The appropriate response to this higher priority aspect of the situation, which could have avoided the stall, would have been to decrease climb angle and accelerate. To quote the investigatory report:

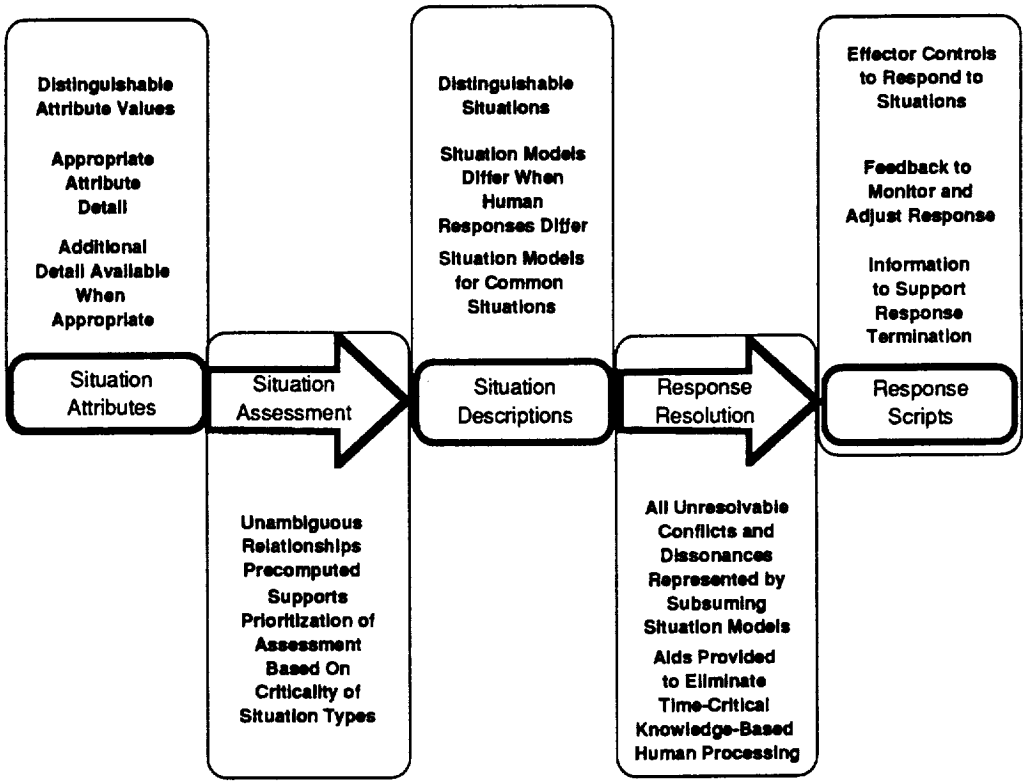


Figure 2-3: Major Steps on the Situation-Response Path

"Each [of these causes: engine loss, slat retraction, warning loss] by itself would not have caused a qualified flight crew to lose control of its aircraft, but together during a critical portion of flight, they created a situation which afforded the flight crew an inadequate opportunity to recognize and prevent the ensuing stall of the aircraft". [NTSB Report #NTSB-AAR-79-17]

The basic problem was that in accepting the initial engine loss situational model, and following the established procedures for situations of that type, the flight crew did not recognize the more critical flight control aspect of the situation.

The challenge for intelligent aiding systems is to be helpful in such emergency situations, when the flight crew doesn't have any resources to spare. Ideally, the aiding system could have prevented the flight crew from accepting the situation as completely described by engine failure on climbout. A properly instrumented situation assessment system would have had abundant evidence that much more than an engine had failed. It is reasonable to assume that a situation

assessment system in the aircraft would have quickly detected the anomalous roll (due to unsymmetrical stall) by monitoring the flight control and inertial systems. An aiding system may have given behavioral advice, such as accelerating, but such unmotivated paradoxical advice might confuse the flight crew. Thus advice such as "Roll Emergency", which implied both the situation and the response, would probably be better. Note that the effectiveness of such communication depends on the flight crew's having models of the low speed roll emergency situation and associated response procedure, and on the effectiveness of the aiding system in stimulating the appropriate pilot situational awareness and response procedures.

2.4.1 Situation-Response and Skill-Based Behavior

Situation-response processing has a role in skill-based behavior. In our model skill-based behavior can be described as the association of behavior with situation attributes directly, without going through the situation assessment process. The pathways in Figure 2-2 for skill-based shortcuts proceed nearly horizontally from the vicinity of sensing and perception to the vicinity of the effectors. The establishment of such skill-based shortcuts reduces workload and reduces processing delay by uncoupling the situation assessment process from the process of adapting to changing situational parameters. In our model, the activation and management of skill-based behavior is one of the normal functions of rule-based behavior. The situation assessment function then assumes the role of enabling, initiating, modifying, and terminating the execution of the skill-based behavior.

Both the risks and speed of skill-based shortcut processing are well known. Pilot interviews provide the following example. During aircraft takeoff roll, the pilot-flying noticed a fluctuation in right engine readings. Just before takeoff velocity the pilot-flying heard a loud boom and felt the plane vibrate. Reflexively he reached to shut down the right engine (having mentally established its potential for failure). The pilot-not-flying stopped this move because he had determined that it was the left engine that had failed. Thus, the processing shortcut (enabled by the assessment of the situation as a possible-right-engine-problem) resulted in a rapid, but inappropriate, response.

2.4.2 Situation-Response and Knowledge-Based Behavior

Situation-response processing also has an important role in knowledge-based behavior. In our model pilot behavior is usually of the situation-response type, with the response procedures involving skill-based components. However, several events can lead the pilot to use knowledge-based processing, namely:

1. Inability to describe the situation in terms of any situation type
2. Inability to resolve the conflicts between the response procedures, or
3. Inability to execute the procedure for the situation, and having no situation types for this execution-failure situation.

Situations of the first type are they are characterized by inability to identify the situation as a whole sufficiently to form a basis for action. A person in these situations normally does nothing, or just continues as before, while trying to understand the situation using knowledge-based processing. Fortunately, pilots are essentially never without some description of the situation. Many pilot situation types (such as flight, cruise, or taxiing) are so general that it takes some really novel perception to violate them all. Typical responses to having no applicable situation types include stopping and thinking about the situation (waiting for more data), or performing some action that will help eliminate or confirm some of the candidate situation types.

The second class (the inability to resolve procedure conflicts) is more important. For example, the inability to resolve the conflict between the flight control emergency procedures and the engine loss procedures may have been a factor in the crash of the DC-10 in Chicago. Resolution of these procedural conflicts may involve knowledge-based activities such as planning, constraint propagation, and means-end analysis. Since the resolution of procedural conflicts is better understood from a machine intelligence perspective, the discussion of the numerous details are deferred to Chapter 3.

The third case (execution failure) is also important in piloting. For example, occasionally the transition of flight control tasks in critical situations can leave the new pilot flying with the impression that the aircraft isn't responding properly to the flight controls. Pilot performance in these situations (which have caused accidents), supports the hypothesis that the pilots do not have situation-response knowledge for these situations. Instead pilots appear to exhibit the signs, such as delays and errors, of knowledge-based reasoning under time pressure.

In all three of these cases, knowledge-based reasoning fills in where situation-response reasoning falls short. The normal pattern is that when one of the three limitations of situation-response processing is reached, knowledge-based reasoning is initiated. Then either the situation changes while the knowledge-based processing takes place, or the knowledge-based processing produces a useful result. In either case the processing reverts to the situation-response model. Note that in this model situation-response processing serves as an input filter for knowledge-based processing, guaranteeing that the scarce knowledge-based processing resources are only invested in unusual, and hence presumably important, problems warranting this most expensive type of processing.

3. A Computational Model Of Situation-Response Behavior

In this chapter we describe a computational model of the situation-response behavior discussed in Chapter 2. An implementation of this model would serve two roles. First, it would support analysis and verification of the cognitive modeling described in Chapter 2. Second, such an implementation, along with the existing implementation of the deep reasoning in the RECORS prototype, would provide a flexible problem-solving system able to perform a wide variety of inferencing types in response to different situation requirements.

The two main tasks that need to be supported by this model are:

- recognizing that a particular situation has occurred (situation assessment), and
- selecting the procedure associated with that situation (response selection).

These apparently simple steps can be made difficult when the assessment function indicates that several situation types appear to be appropriate candidates to account for the available data and the pilot must either choose one, in order to select one response procedure, or combine several procedures in order to react to the several situations. The role of an aiding system becomes particularly important in situations where the pilot becomes overwhelmed by the data or misclassifies the situation.

The remainder of this chapter will discuss in detail the type of knowledge and reasoning required for the situation-response model. The description will follow the information processing stages in the model, moving step by step from the perceived data (situation attributes), through situation assessment (situation typing), situation-response association, and finally to response resolution. We will also discuss in detail the points in the processing where ambiguity arises, in both the situation assessment and the response selection.

3.1 Situation Attributes

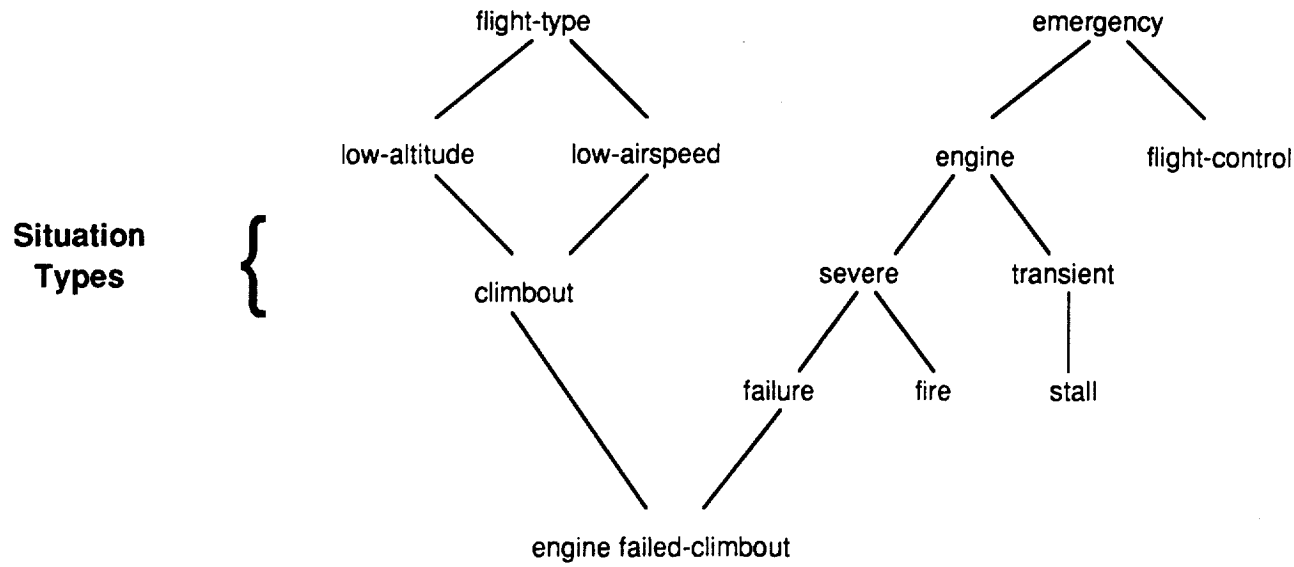
We begin the more detailed explanation of the computational model for situation-response behavior with the stages of late perception - early situation assessment. In these stages the available data have not been processed to the point where a known situation has been recognized which could initiate the situation-response behavior. We call these initial, raw data, the SITUATION ATTRIBUTES. These include dynamic information from the airplane sensors (oil pressure, engine status), information about the particular flight (flight phase, route), and background knowledge such as the weather or the specific airports involved. The situation attributes comprise the input to the aiding system.

3.2 Situation Types

Situation types are attempts to represent the pilot's mental models of classes of situations at the situation-response level of abstraction. The situation attributes are one step below the situation types in the abstraction hierarchy. Figure 3-1 shows a portion of a situation hierarchy, focusing on engine failure emergencies. In terms of the frame representation often used to express these types of hierarchies, situation types are individual frames and situation attributes are the frame attributes or slots that characterize each frame.

The basic difference between situation types and situation attributes is that response procedures are associated with situation types but not with situation attributes. The scope of the situation attributes is not large enough to permit the direct association of a particular behavior. In other words, an individual situation attribute does not contain enough information to indicate what the appropriate response should be. The situation attributes do however influence the situation-response behavior indirectly, through their influence on the selection of the appropriate situation type, and through the execution of the associated response procedure.

Situation types in both the aiding system and pilot model should be different for distinguishable situations which require different types of responses. Thus, if two situations which are distinguishable by observable situation attributes have different responses, then those



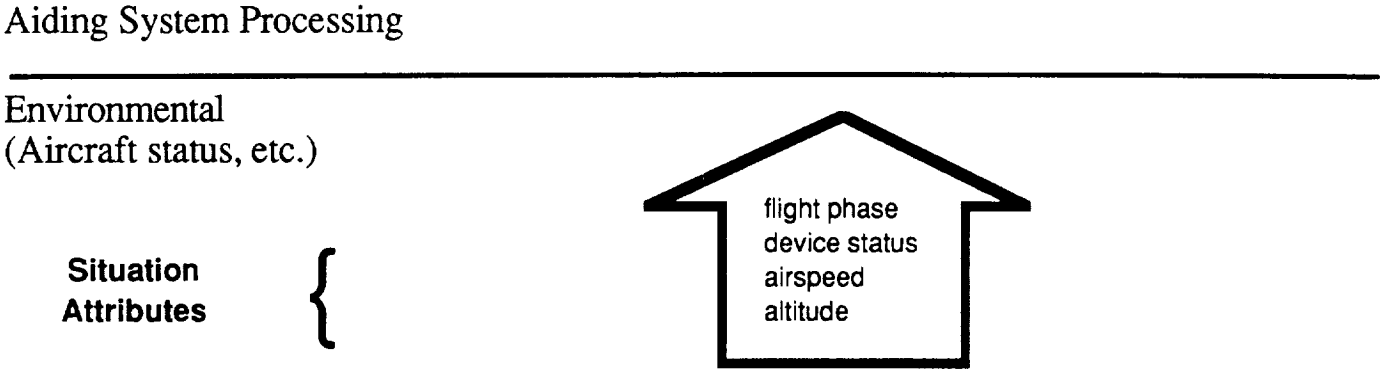


Figure 3-1: Relationship Between the Situation Attributes and the Situation Type Hierarchy

two situations should belong to distinct situation types. Conversely, if two situation types are never distinguishable based on their attributes, then the two types should be combined, and

behavior associated with the combined type should be appropriate for a situation which could be of either type. Similarly, if two situations may or may not be distinguishable, based on situation attributes, and those situations have different behaviors, then it is often appropriate to generate a more general situation type which models the uncertainty by spanning the set of indistinguishable situation types. The behavior associated with this more general situation type should be appropriate for the state of knowledge of the situation. When more specific situation classification is possible, then the appropriate, more specific situation type, should be used.

Examples of situation types are ENGINE-FAILURE-CLIMBOUT, FLIGHT-CONTROL-EMERGENCY, and ENGINE-FIRE. The distinct situation types are represented by *frames*. A frame is a representational structure consisting of a collection of attribute-value pairs which together describe some object, such as a situation type or a response procedure. (The attribute-value pairs are also called slots and slot values (fillers) in the AI literature.) For example, the frame representing the situation type ENGINE-FAILED-CLIMBOUT[3] includes the following attribute-value pairs:

```
Flight Phase:    climbout
Engine Status:   failed
Airspeed:        V2 +5
Location:        wing left
```

In a frame representation system we distinguish among two types of frames: uninstantiated and instantiated. An uninstantiated frame represents a generalized description of some object.[4]Because such a description must encompass a larger class of objects, some of its attribute values are underconstrained. Thus an uninstantiated frame representing the general class of ENGINE-FAILED-CLIMBOUT situations might have the following attribute-value pairs:

Flight Phase: climbout
(This attribute is fully constrained in the uninstantiated frame
since here we only represent climbout failures.)

Engine Status: failed

---

[3]Unless otherwise indicated, all references to aircraft assume a DC-10 aircraft, with two wing-mounted engines and one tail engine.

[4]The term "object" is used in the sense of a data item, it can represent a physical object, a situation, or some system state.

(Engine Status is again fully constrained because this
situation describes an engine failure)

Airspeed: V2 to 200 knots
(The airspeed is not known until the data arrives, it is
however, constrained by the range of speeds consistent with that flight
phase.)

Location: wing left|wing right|tail
(This attribute is underconstrained to the three possible engine
locations since we don't know which engine failed until the
data arrives.

This frame thus describes the general engine climbout failure situation but does not fully constrain all attributes. The "Airspeed" and the "Location" attributes cannot be determined until the corresponding situation attributes become known. One generalized situation description can thus be instantiated to represent a large number of specific situations. This instantiation process consists of constraining (or refining) the attribute values and comprises the processing during the SITUATION ASSESSMENT phase which is described later. For example, if engine 1 failed, the "Location" attribute value would become "wing left" and the "Airspeed" attribute would become whatever the current air speed was. An instantiated situation type thus corresponds to a specific ENGINE-FAILED-CLIMBOUT situation.

In many cases the objects represented by the frames are related in some way, for example, by a subclass or superclass relation. The SEVERE-ENGINE-EMERGENCY situation is a subclass of a more general situation: ENGINE-EMERGENCY. This is represented by having the value of SUPERCLASS attribute point to another frame. The SEVERE-ENGINE-EMERGENCY frame thus contains the attribute-value pair,

Superclass: ENGINE-EMERGENCY

Since frames can point to other frames the resulting structure is a type of a graph. In most cases it is a semi-lattice,[5] where the leaves are the most specialized objects and the roots the most abstract.

[5]A semi-lattice is a graphical representation of a partially ordered set which has only a least upper bound or greatest lower bound but not both.

An advantage of this type of structure is representational efficiency because shared attributes need only be represented once, at the level of the highest object they apply to, and are "inherited" as necessary to the more specific ones. In our example, the ENGINE-FAILED-CLIMBOUT frame thus consists of its own attribute-value pairs, the attribute-value pairs of its superclass situations, and so on, recursively, until a situation is reached which has no superclass. Figure 3-1 shows a portion of the structure describing the different situation types involving engine failure at takeoff and the relationships among them. This type of representation is common in classification-type problems, of which situation assessment is an example.

3.3 Situation Response Procedures

Associated with each situation type is a description of the response procedure to perform when that situation occurs. In engine failure situations there is a fairly limited, prescribed set of responses and one response may be used in several different types of situations. For example, in the ENGINE-FAILURE situation the response is the Engine-Failure procedure. In our frame representation this means that each situation frame has an attribute called RESPONSE-PROCEDURE, whose value is the name of the frame containing the description of the response procedure. More than one situation type can point to the same response procedure. Responses are associated with the situation types as incompletely specified response procedures, parameterized with respect to the specific attributes of the situation types: the situation attributes. When situation assessment refines the situation attribute values and instantiates a particular situation type, the response procedures are also refined. Thus the response procedures are instantiated with respect to a particular situation much as the situation types are.

For example, part of the response procedure for engine failure is to perform an electrical power check. Depending on the number of failed generators, the pilot has to reduce the power consumption to different levels by shutting off different sets of electrical subsystems.

There may be cases when the situation assessment process cannot identify a unique situation to account for the data. In these cases multiple responses may seem appropriate and the inferencing mechanism must select among them or combine them. This process discussed in more detail in Section 3.5.

3.4 Situation Assessment

Situation assessment is the process of mapping the situation attributes into the appropriate situation types, resulting in the **instantiation** of a particular situation type. Instantiation refers to the process of assigning specific values to the attributes of the situation type frames.

The situation assessment process can be modeled computationally by a matching or taxonomic classification process, where the perceived situation attributes form a pattern and the goal is to find all situation types that fit that pattern. The situation attributes are thus the leaves of the taxonomic lattice discussed in Section 3.3. During situation assessment the attribute values of the situation frames are constrained by the actual sensed data (the situation attributes). For example, if the altitude fluctuates between 50 and 150 feet, then only situation types satisfying this constraint need be considered as candidates for the correct interpretation of the current data.

3.5 Ambiguities in Situation Assessment

We will now discuss the adequacy, in terms of completeness and correctness, of the situation assessment process. Ideally, all possible situations could be described in terms of situation types that would account for all observed data. Such a well-fitting situation type would then result in an equally appropriate situation description and a well focused response procedure.

In some domains, tailoring situation types to each of the combinations of the situation attributes is possible because the sensor data are sufficiently accurate, and the number of situation types, and the combinations of different situation attributes, are sufficiently small. However, the number of different combinations of situation attributes precludes the unique association of a situation type with each possible attribute combination in most realistic domains. This gives rise to ambiguities in the situation assessment process, because more than one situation is triggered by a given set of attributes. This is a common problem in AI systems. We will adopt the established terminology of rule-based (RB) expert systems, where the candidate situation types (rule predicates, in the case of RB expert systems) comprise the **conflict set**, and the process of selecting a unique situation type is called **conflict resolution**.

The ambiguities can be divided into two categories.

- UNCERTAINTIES IN THE DATA lead to uncertainties in the situation attributes. For example, a particular sensor reading might not provide a specific value but rather a range of values - e.g., an allowable variance in engine oil pressure of plus/minus several percent of the nominal value.
- The structure of the TYPE TAXONOMY (which is NON-DETERMINISTIC due to the complexity of the domain) leads to ambiguities in situation assessment, since one set of situation attributes may be consistent with several situations.

3.5.1 Resolving Ambiguities in Situation Assessment

We will consider several techniques that help manage the non-determinism in situation assessment resulting from inadequate or noisy data. The constraints and advantages of applying these techniques are described below.

1. Combining data-directed and goal-directed processing when instantiating the situation frames.
2. Using a more general situation description, which covers all candidate situation types.
3. Increasing the granularity of the type taxonomy by adding more generalized situation descriptions.
4. Refining the granularity of the type taxonomy by introducing subtypes which describe different aspects of situations.
5. Assigning fixed priorities to the situation types and selecting the situation with the highest priority.

The first two of these techniques involve more inferencing during the situation assessment and thus impacts the architecture of the system. The next two involve changes to the knowledge-base, in this case the taxonomy of the situation types and must be performed during the knowledge acquisition stage.[6] The final approach involves additional knowledge. We will discuss each of these techniques in more detail below, pointing out their advantages and limitations.

[6]It is also possible to integrate knowledge acquisition with processing, by having the system adapt its knowledge-base structure.

3.5.2 Combining Data-Directed (bottom-up) and Goal-Directed (top down) Processing

The most basic technique for handling ambiguity is to make full use of both the sensed situation attributes and the constraints provided by candidate situation types. This simply means that we are using all the available knowledge; both the dynamic knowledge, as represented by the current data, and the more long-term knowledge, as represented by the situation type taxonomy. This technique relies on the fact that the attribute values in the instantiated situation descriptions must be compatible with both the perceived data (situation attributes) and the constraints placed on the attribute values by the uninstantiated situation types (see Figure 3-2). The uncertainty in the data can at times be resolved, or at least reduced, by combining the constraints provided by the data (data-directed processing) with the constraints provided by the candidate situation types (goal-directed processing). Similarly the uncertainly due to a non-deterministic type taxonomy can at times be resolved by the existing data. This combination of data-directed (bottom-up) with goal-directed (top-down) processing is a common AI technique for handling uncertainty [Corkill 1982].

The top-down constraints can also be used to direct the gathering of more information about the attributes of the situation. This additional perceptual information may further constrain the values of the situation attributes. This in turn may resolve any inconsistencies by eliminating some of the inconsistent situation type candidates.

3.5.3 Resolution by Taxonomic Generalization

Subsumption relations between situation descriptions can play a key role in resolving the ambiguities of assessing situations with incomplete information. If two situation descriptions are inconsistent, they may both be specializations of a more general situation description; i.e., the more general situation type is a superclass of both. This is the case anytime the situation types are arranged in a taxonomic semilattice and it is one of the primary reasons for structuring the situation types in a semilattice. For example, if both an engine failure and engine-fire seem to explain the current data, then their superclass, SEVERE-ENGINE-EMERGENCY, may be selected (refer to Figure 3-1).

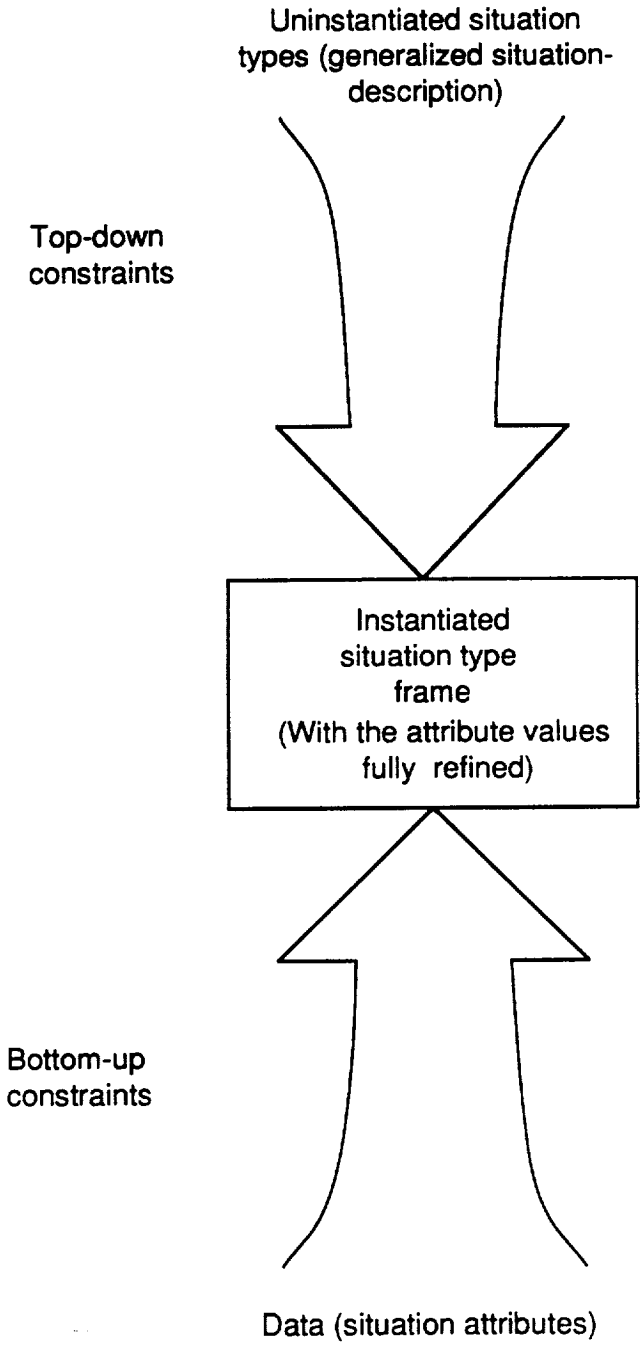


Figure 3-2: Use of Bi-directional Constraints in Frame Instantiation

The disadvantage of this approach is that the procedure associated with the more general situation type may be too general to be useful, since the procedures for the general situations typically include more variations and flexibility than procedures for more specific situations. Obtaining appropriate responses from such general situation descriptions will therefore require more computation during the procedure refinement stage. This approach thus tends to move the computational burden from the situation assessment stage to the response resolution stage of processing. An additional problem here is the necessary loss of information as we move up the taxonomic hierarchy, since some of the situation attributes were necessarily not used in the instantiation of the more general situation type.

This approach has its own sources of ambiguities. Suppose that one of the candidate situations has two superclasses. We cannot, therefore, identify a unique superclass of both candidates. We have two choices: we can select the intersection of their superclass at the lowest level or we can go to a level where there is no ambiguity. The danger with the second approach is that a situation that accounts for all candidate situation types may be too general to be useful.

3.5.4 Increasing the Granularity of the Type Taxonomy

The generalizing technique described above only works if the conflicting situations have a common superclass of reasonable specificity. When the need for this type of conflict resolution arises often and fails, it may mean that the type taxonomy should be modified; that new, more general situation types should be added. These can be tailored to resolve the more commonly occurring conflicting situations. The problems with this approach are:

- a proliferation of situation types in the taxonomy which leads to an unwieldy knowledge-base, and
- the difficulty of constructing an appropriate response procedure which would combine the requirements of all the conflicting situations.

3.5.5 Refining the Granularity of the Type Taxonomy

Another approach to conflict resolution is to capture different aspects of a situation in multiple, more limited descriptions of the same situation. This technique, again involving changes in the knowledge-base structure, is the opposite of the one described above. Here the scope of what a situation type describes is decreased by including fewer attributes. Such partial situation descriptions thus describe only aspects of a particular situation. This approach also involves changes in the processing architecture, since here it is desirable to have multiple situation aspects triggered to account for the existing data. We no longer need to reduce the conflict set to just one situation, because each partial situation description has its own response procedure. An example of such a classification would be to have engine failure and loss of thrust at low altitude as two separate situations instead of loss of engine at low altitude.

The response procedures associated with the situation aspect types describe only that portion of the behavior which can be inferred from the subset of the situation attributes that matched. Different aspect types can match different subsets of the situational attributes, resulting in several partial situation descriptions and different partial response procedures, which then have to be combined. The difficulty of combining the partial procedures can range from trivial, when the procedures do not interact, to impossible, when the procedures have irreconcilable conflicts.

3.5.6 Note on the Granularity of the Type Taxonomy

There is a tradeoff between using more abstract situation types (large granularity) and partial situation descriptions (small granularity). General descriptions typically require more processing and hence longer delays than descriptions in terms of situation aspects. For example, an indication of engine fire may suggest the immediate discharge of a fire bottle. This is a limited response to a limited description. A broader assessment of the situation might suggest that assured incapacitation of the engine through fire bottle discharge should be avoided until other options are considered and the best response to the full situation is identified. The benefit of response based on limited aspects of a situation is speed, at the cost of eliminating options. The cost of the generalized response is computational time and complexity in a critical situation.

(This tradeoff is analogous to the one between shallow and deep reasoning.) This represents a type of a problem where real-time constraints come into play.

3.5.7 Fixed Situation Priorities

Conflict among competing situation type candidates can also be resolved by *a priori* assigning fixed priorities to each situation type. When ambiguous data cannot be interpreted by a unique situation, the situation with the highest priority has precedence. For example, a situation involving fire might have priority over any other situation during the cruise flight phase. This priority assignment will not hold over to the entire set of the situations but only for situations within each flight phase. The priorities are thus context sensitive with respect to the flight phases.

3.5.8 Using the Above Techniques to Resolve Ambiguity

The combination of data-directed and goal-directed processing is a part of the situation instantiation process. This technique is applied first in order to fully exploit both the information that is already in the knowledge-base, and the current data. This technique also has the potential of completely disambiguating the candidate situation.

If this method fails, the inferencing mechanism selects the situation with the highest priority. If this fails to identify a unique situation then the taxonomic generalization is used. The complementary techniques of increasing or decreasing the granularity of the situation type taxonomy can only be applied after the system has been running for some time and the effectiveness of the current taxonomy can be measured.

3.6 Ambiguities in Response Resolution

The same ambiguous situations that can occur in the absence of complete information during situation assessment can also occur during response selection. This type of uncertainty can be the result of:

- the inability of the situation assessment procedures to select a unique situation to account for the data, or
- the use of multiple situation aspects in describing a situation, and the consequent set of available partial response procedures.

The first ambiguity is illustrated by one pilot's inability to immediately distinguish between an elevator failure and a compressor stall on a failed tail engine as illustrated by the example below.

The aircraft was in the cruise phase of flight and was experiencing severe buffeting, when the pilot heard a loud bang, followed by the nose pitching up. The pilot assumed that the elevators had failed, causing the change in pitch, because he did not at that moment realize that a stalled tail engine would also have caused a pitch up. He slowed down to normalize the pitch and then discovered that the elevators were in fact functioning, at which point he realized there had been a stall and successfully restarted the engine.

We have not encountered the second type of ambiguity in the flight domain but are discussing it for the sake of theoretical completeness. The technique described below can handle both situations.

The techniques for conflict resolution among the potential responses are similar to those used in resolving the conflict during situation assessment. These are:

- combining procedures,
- using a more general procedure, and
- using a least commitment response and waiting to see if the situation "takes care of itself".
- waiting to see if more data becomes available over time.

(Note that we do not include the prioritizing of response procedures as a conflict resolution technique, because if this technique was possible, it would be possible at the level of situation prioritizing, and the uncertainty would never reach the stage of response selection.)

3.6.1 Using a More General Procedure

This technique is analogous to the one described in situation assessment conflict resolution. Like the situations, the response procedures can be organized into a hierarchy. When two procedures seem appropriate, a more general procedure, which is the superclass of the two, can be executed. For example, suppose the pilot is not sure whether the hydraulic fluid quantity or the pressure is abnormal. He can then execute the procedure for hydraulic system system failure, which is at a higher taxonomic taxonomic level than either of the more specific procedures.

This hierarchical taxonomy can be adaptive and frequently occurring resolutions can be compiled into new situation-response pairs which subsume the situation-response types that have previously needed to be resolved. Thus both the type taxonomy and the situation-response mapping can be refined with use.

3.6.2 Using a Least Commitment Response

Inconsistencies in the responses need only be resolved when the inconsistent parts of responses actually need to be executed. For example, different engine problems at takeoff may eventually require different responses, depending on the diagnosis, but the responses are all the same (leave the engine running) until the aircraft has attained safe altitude and air speed. By proceeding on the consistent response subset, and deferring the inconsistent response subset, the inconsistency may disappear, as new information is gained during response execution. Sometimes the whole response can be deferred until the inconsistency is resolved.

Deferring tasks requires reasoning about task scheduling and time constraints. For example, engine fire-bottle discharge might be deferred in an abortive take-off until reverse thrust from that engine is no longer required.

4. Architecture of RECORS

4.1 Introduction

Overview. In this chapter we describe in detail the architecture and functionality of the Recovery Recommendation System (RECORS). RECORS is not a stand-alone system intended to support automated flight but is rather a first step towards providing an integrated intelligent interface in the cockpit. The aim of RECORS is to provide some of the problem-solving capabilities necessary to provide intelligent assistance to the crew and to support an intelligent interface. Such interface would:

- display only information relevant to the situation at hand,
- display this information at the appropriate level of abstraction so as to minimize the cognitive load on the pilot, and
- help with the data-rich but information-poor situation pilots have to face, particularly during emergencies.

We think of RECORS as an intelligent agent serving as a mediator between the large amounts of raw data available from the aircraft monitors and the pilot. The aim of this agent is to monitor the data, assess the situation and inform the pilot of the important features, and suggest appropriate responses. If necessary, this agent should also be able to provide explanations for its answers.

The RECORS prototype will be performing similar tasks, and processing the same data, as the pilot, throughout the duration of the flight. We believe this parallel execution is necessary in order for the system to provide the pilot with relevant advice in a timely fashion. Figure 4-1 illustrates this parallel execution and the possible places of interaction between the pilot and RECORS.

Scope of RECORS and its relationship to FaultFinder. The RECORS prototype is intended to be integrated into the series of pilot aiding systems currently under development at NASA-Langley [Abbott, 1987, 1988]. (See Figure 4-17.)

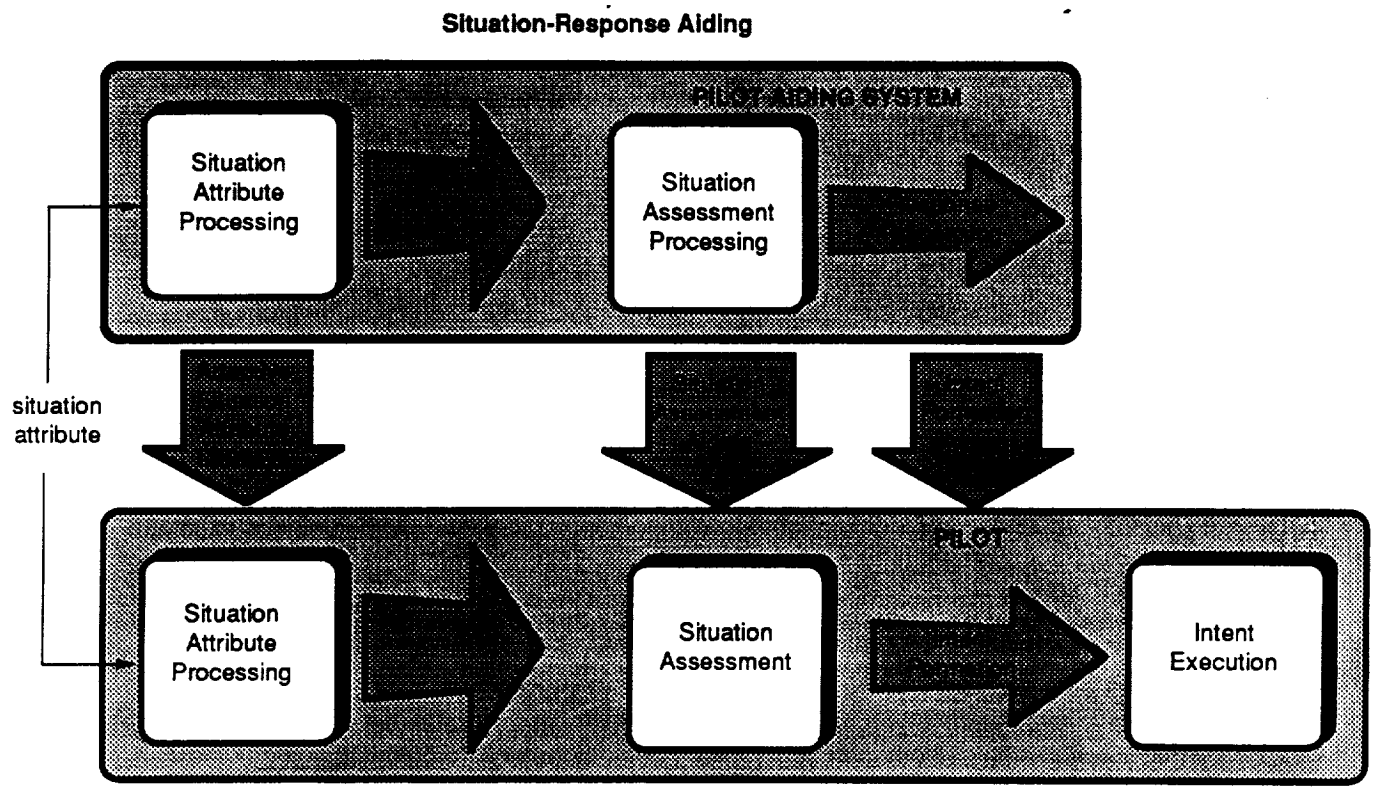


Figure 4-1: Points of Interaction Between the Pilot and RECORS

RECORS receives its input from the FaultFinder in the form of faulty aircraft components and produces as output:

- a list of affected flight profile characteristics (airspeed, altitude, aircraft attitude),
- specific alarms or warnings (aircraft stall, slat disagree warning, etc.), and
- suggested responses to stabilize the situation.

The aircraft currently modeled by RECORS is the DC-10, with two wing-mounted engines and one tail engine.

Design features: causal model and multiple abstraction levels. The design of RECORS's knowledge bases and inferencing structure was driven by two considerations:

- that it facilitate simulation of multiple failures and derivation of responses from first principles, and to that end that it represent causal knowledge of the flight domain,
- that it integrate with the architecture of the FaultFinder diagnostic system, and be able to process the FaultFinder's output.

The first consideration led to the use of a causal model, which represents the causal relationships among the various aircraft subsystems, effectors, and flight characteristics. In order to capture the underlying principles of the flight domain, the model explicitly represents the major forces that act on the aircraft. This representation combines interacting causal pathways, making it possible both to simulate multiple failures and to derive multiple responses to achieve some desired state.

The second consideration led to the introduction of the binary and qualitative levels of abstraction. At the binary level we distinguish between normal and abnormal states or behaviors. This level makes it possible to make inferences in the absence of detailed information about the aircraft. Reasoning at this level is typically based on propagation of physical malfunction data in the FaultFinder which, necessarily, cannot be very exact, and corresponds to the "affected" value produced by the FaultFinder. For example, if an engine fails, the physical propagation model may predict that the surrounding control surfaces will be affected. The RECORS system can then propagate these effects forward through in the model to see how they will impact the flight characteristics. Inferences at this level are not specific enough to support stand-alone functionality. However, since RECORS's ultimate purpose is to filter out or highlight information for presentation to the human flight crew, even very high level inferences may help to focus the crew's attention to a problem area.

At the qualitative level, the state and behavior are described by one of several possible qualitative values. Typically there are three such values, corresponding to a stable state and the changes from this state in two directions. For example, the qualitative values describing airspeed are: decreasing, stable, and increasing. The qualitative level thus requires more information than the binary level and provides correspondingly more detailed inferences. Since the number of qualitative values is small, it is possible to perform a complete search through the qualitative space even when only binary-type information is available. For example, knowing only that an "engine has been affected", RECORS can simulate the effects of both a decreasing thrust and increasing thrust. The possible outcomes are displayed to the pilot, who can then use other information to decide which is in fact the case. Orthogonal to the binary and qualitative levels are the abstractions defined by the taxonomic organization of the model. These levels support reasoning about individual aircraft components (left or right engine) or about the engines in general. Again, these abstractions support reasoning in the absence of detailed information about the state of the aircraft. The details of how reasoning at the different abstraction levels is integrated are discussed in Section 4.3.2.

Quantitative Modeling. The original design of RECORS provided for a third, quantitative, level of abstraction [Hudlicka, 1988]. At this, the most detailed level, the flight domain would have been represented by the dynamic equations of flight. We have postponed the implementation of this level because we wanted to investigate the use of existing quantitative models such as those present in the on-board flight directors and training flight simulators to support quantitative reasoning. In the next phase of this project, we plan to incorporate these existing models into RECORS, rather than duplicating the work that has already been done.

Architecture overview. RECORS is implemented in the KEE™ 3.1 development environment on a Symbolics 3600 series under Genera™ 7.1. The knowledge representation scheme is mixed, consisting of frame taxonomies, qualitative constraints, causal links, and procedures. The declarative knowledge is represented by frame taxonomies describing:

- a causal model of the flight domain, representing the interactions among the various aircraft subsystems, effectors, forces, and the flight profile characteristics,
- specific undesirable situations (alarms and warnings), represented as constraints among the attributes of objects in the causal model, and

- desired effector settings and flight profile for each flight phase, represented as constraints on the attributes of the effector and flight profile objects.

The causal knowledge is represented by linking the causally related objects within these taxonomies. These links exist both at the binary and at the qualitative levels. At the qualitative level they consist of qualitative constraints which express the nature of the causal relationship and support both failure simulation during situation assessment and the derivation of alternate responses during response aiding. These knowledge bases are discussed in Section 4.2. The procedural knowledge consists of the inferencing mechanisms written in LISP. These are:

- simulation of a faulty system or effector,
- triggering of warnings and alarms,
- determining which flight phase goals have been compromised,
- selecting a response action, and
- determining how this action should be achieved, given the constraints of the faulty components and the current flight phase goals.

These mechanisms will be discussed in Section 4.3.

Chapter Organization. The remainder of this chapter is organized as follows. Section 4.2 describes the structure of the knowledge bases. Section 4.3 describes the inferencing mechanisms which use the knowledge to perform the processing. Section 4.4 describes the interface design and user interaction with the system. Section 4.5 discusses directions for future work.

4.2 System Architecture: Knowledge Bases

4.2.1 Model Structure

Overview. The core knowledge base of RECORS is a model representing the causal relationships among the aircraft subsystems (fuel, hydraulic, oil), the effectors (engines, control surfaces), and the flight profile characteristics (airspeed, altitude, roll, pitch, yaw) (see Figures 4-2 and 4-3).

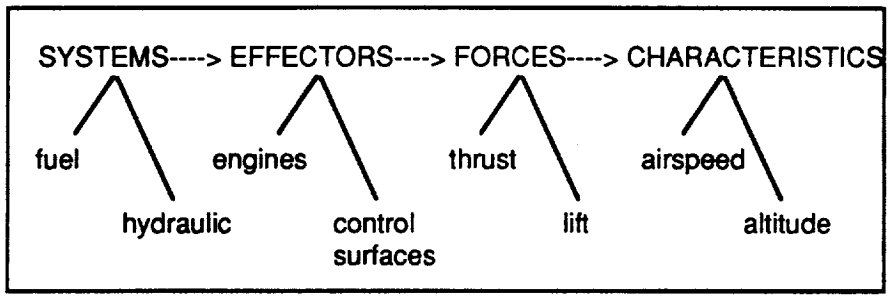


Figure 4-2: High-Level View of the Causal Model Showing both the causal links and the Taxonomic Links

There are two major causal chains in the model: one representing the power (linking engines to thrust to airspeed) and the other representing the control (linking the control surfaces to the lift and side forces to roll, pitch, and yaw) (see Figure 4-4).

The model tries to capture the principles of the domain by explicitly representing the major forces that act on the aircraft in flight: lift, thrust, drag, side force, and gravity. Since it is these forces that directly affect the flight profile characteristics, rather than the effector settings, their explicit representation makes it possible to combine the influences of several effectors. This becomes useful both when little information is available about the state of the aircraft and in generating multiple responses to achieve a desired flight characteristic. For example, a roll is caused by asymmetrical lift on the two wings which can be achieved by several effectors: the preferred method is to use the ailerons, but asymmetrical slats, flaps, or spoiler settings can also lead to a roll. By representing the underlying force, we can combine both normal (ailerons) and abnormal (asymmetrical flap, slat or spoiler setting) conditions. This is particularly useful when the normal way of inducing a roll is not possible (e.g., ailerons are non-functional) and an alternative method must be found.

ORIGINAL PAGE IS
OF POOR QUALITY

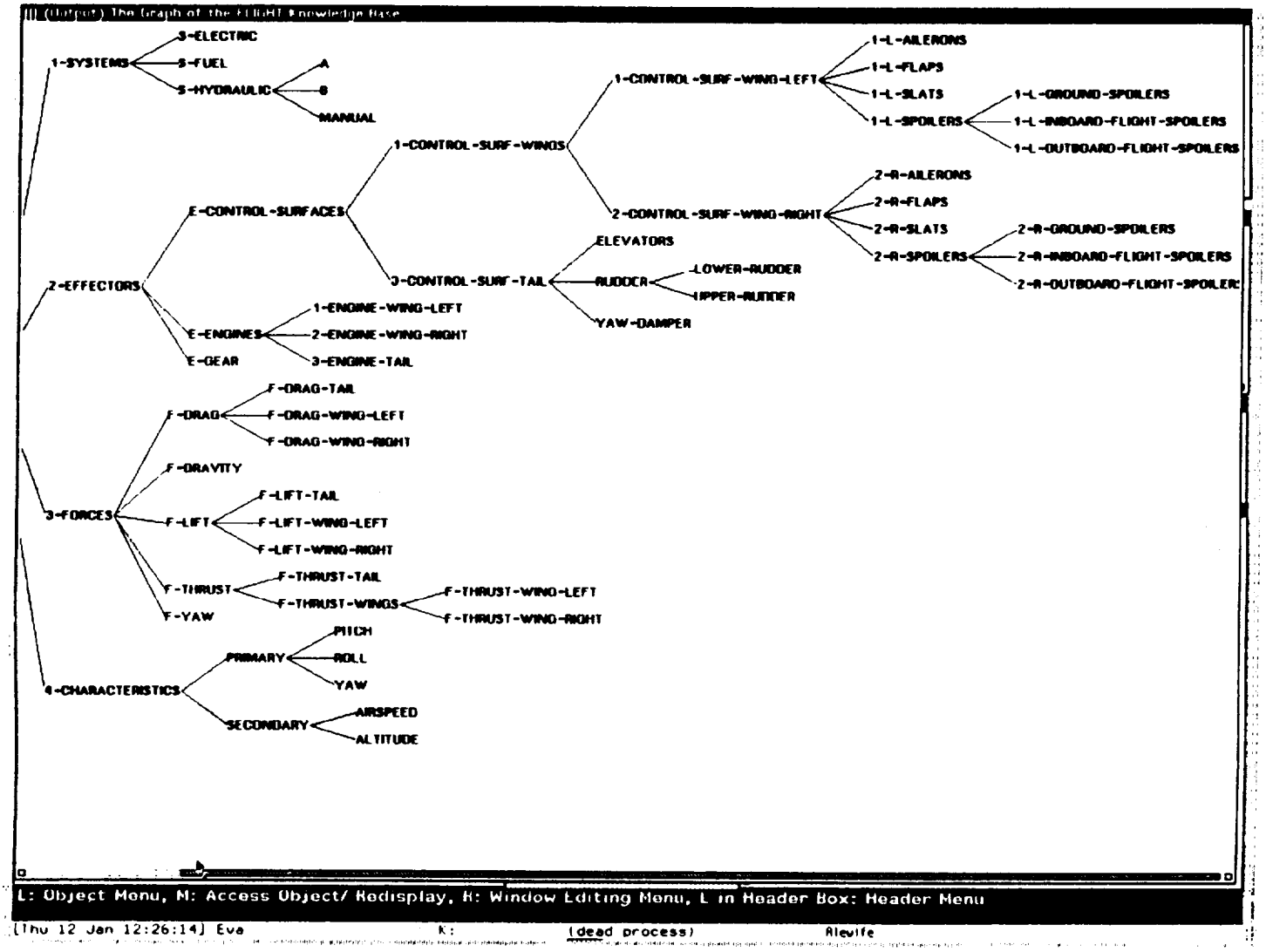
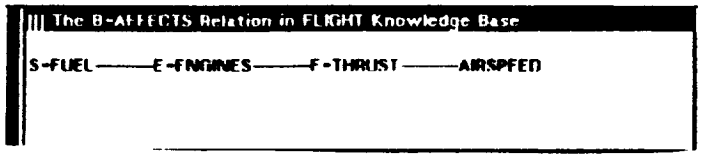


Figure 4-3: The KEE Representation of the Core Causal Model. (Only the IS-A Links are Shown.)

a. **Power Path**



b. **Control Path**

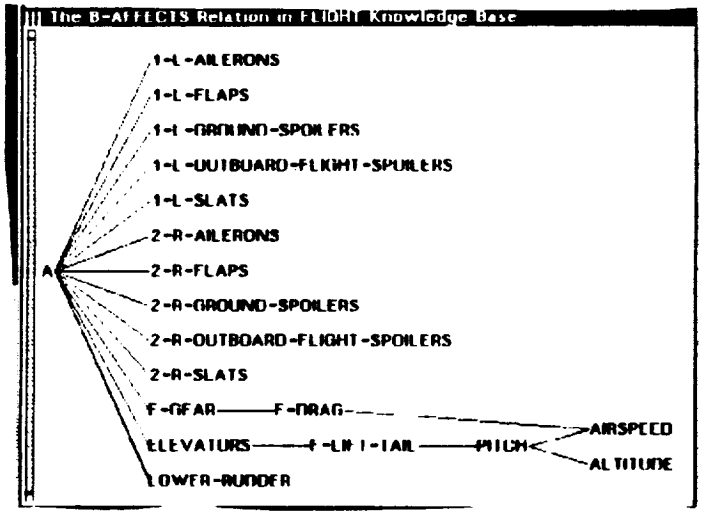
control-surfaces-wing-left → lift-wing-left

control-surfaces-wing-right → lift-wing-right

elevators → lift-tail → pitch → airspeed

rudder → f-yaw → yaw

c. **Hudraulic System/Control Surface Links**



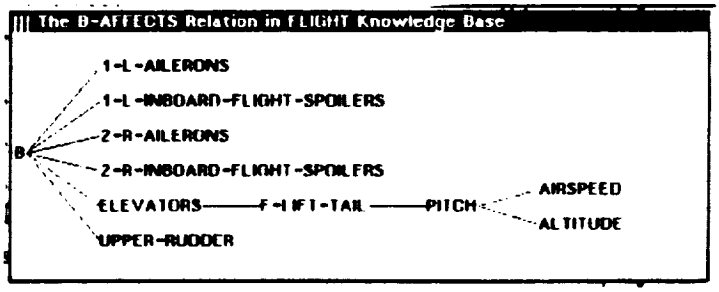
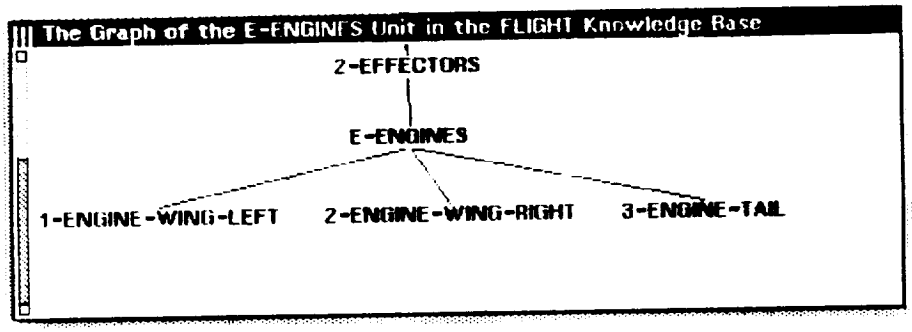


Figure 4-4: The Major Causal Paths in the Model, Representing Power and Control

Model Objects. The model consists of objects represented by frames (units) which refer to physical parts of the aircraft, the forces, the flight characteristics, or to groups of these represented by a unique object in order to facilitate some inferences. Examples of such groupings are shown in Figure 4-5.



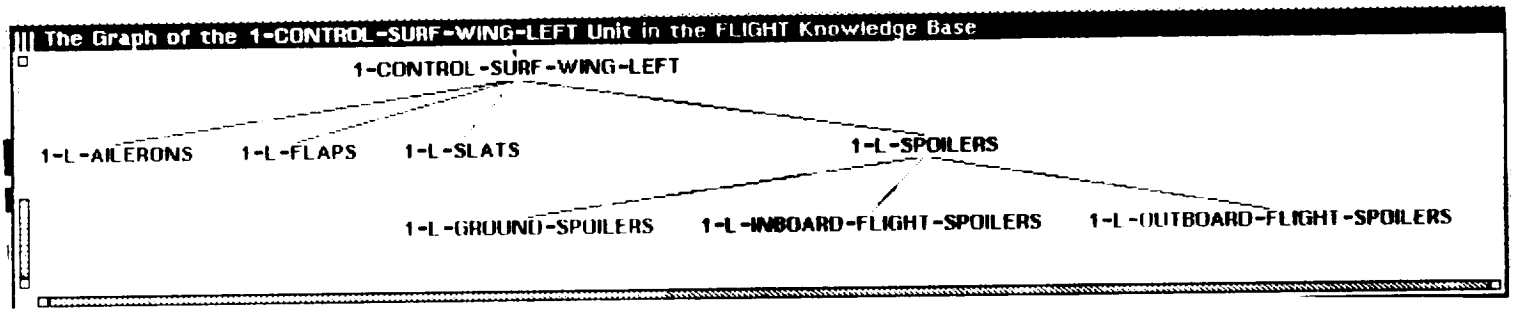


Figure 4-5: Examples of Grouping Together Related Effectors Under More Abstract Categories.

The objects' attributes (slots) represent the current state of the object (the status attribute), the direction in which this state is changing (the trend attribute), and the object's relationships to other objects (the affects, affected-by, parent, and child attributes). Figure 4-6 shows the object structure. These objects form both a taxonomy (defined by the parent-child links) and a causal model (defined by the affects/affected-by links). The model thus combines both causal and

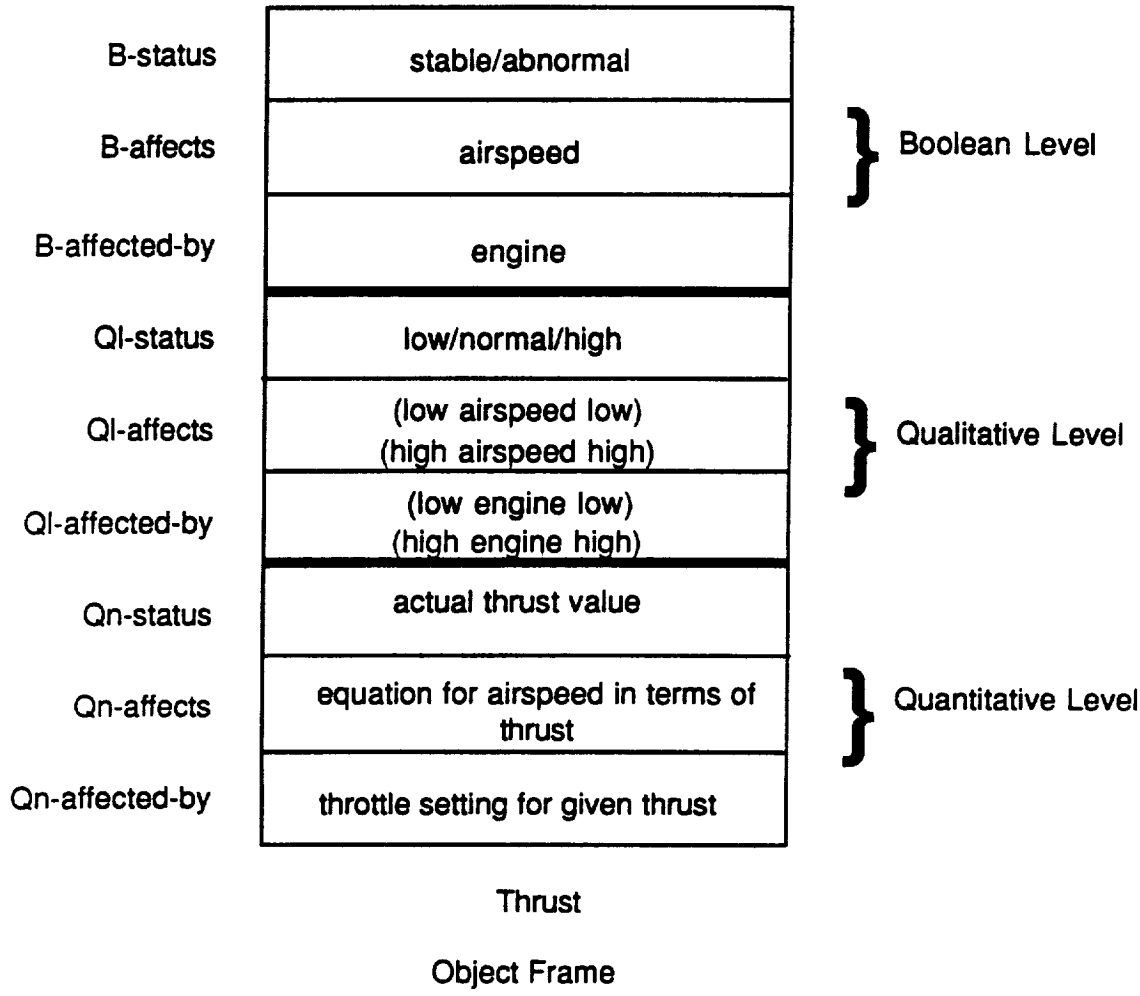


Figure 4-6: A Subset of the Object Attributes Showing the Status and Causal Links at Three Levels of Abstraction.

taxonomic (class membership, taxonomic) relations in a single structure. The issues associated with such a combined representation are discussed in Section 4.2.5.

Levels of abstraction. Each object's status, trend, and causal link attributes exist at both the binary and the qualitative levels of abstraction. Only the status attribute is used at the binary level since the information contained in the trend and status attribute at this level is the same. See Section 4.2.2 for a detailed discussion.

4.2.2 Object Description Attributes

Overview. The state of each object in the model is represented by two attributes: status and trend. The status attribute represents the current state of that object; the trend represents the direction in which this state is changing. The differences between these two attributes with respect to causal and qualitative reasoning are discussed below. In addition to the status and trend attributes, the objects representing the aircraft systems and effectors also have an attribute indicating whether the object is functional; i.e., whether its state or behavior can be changed. This attribute is necessary both to simulate faults where an effector is stuck in a particular state and to eliminate responses which involve non-functional effectors. First, a brief definition of each attribute is provided and then the issues associated with the use of the different attributes are discussed.

Status. The status attribute represents the current state of the object. There is no indication as to the duration or the stability of that state. At the binary level, the status can only be normal or abnormal. At the qualitative level the status can take on one of several values which describe the possible qualitative states of the object. These values are different for the different objects. For example, the qualitative values for an engine are: off, idle, low, medium, high, max. For control surfaces these values are: raised, or lowered (spoilers); extended, or retracted (flaps, slats); left, straight, or right (rudder).

Trend. The trend attribute represents how the current state is changing at the particular instant. It does not represent the behavior of the object over time. At the binary level, the trend attribute is again normal or abnormal. Thus at this level of abstraction there is no difference between the status and trend attributes. At the qualitative level, the trend attribute corresponds to the sign of the first derivative of the object's state. As with status, it can take on a variety of values, depending on which object is being described. Typically there are three values, corresponding to a positive change, negative change, and a stable state. For example, engine, thrust, airspeed, and altitude can be either increasing, stable, or decreasing, resulting in the familiar set of qualitative values denoted -, 0, and + [de Kleer, 1985, Forbus, 1985]. For control surfaces affecting the lift, the qualitative values are: raising, stable, and lowering. For the rudder, the control surface affecting the yaw force, the values are left (meaning "going to the left"), straight, and right; similarly, yaw itself has the same set of values.

Functional? Unlike the state or behavior attributes, the *functional?* attribute provides some indication of time and the object's state and behavior over time. Namely, it indicates that the object is not functional and thus its state (and behavior) will not change. The *functional?* attribute is associated with each subsystem (fuel, hydraulic) and each effector (engines, control surfaces). It can be t or nil, corresponding to a functional or non-functional system or effector respectively. The nil value thus indicates that the current state of this system or effector cannot be changed. This attribute is needed in order to capture "stuck-at" failures; for example, flaps which are stuck in either the extended or the retracted position.

4.2.3 Differences Between Status and Trend

Although the status and trend attributes are closely related, there are several important differences between them with respect to:

- representational resolution,
- qualitative arithmetic, and
- representation of time.

These differences are discussed below by way of explaining our choice of the trend attribute as the basis for causal reasoning.

We have also included the status attribute in the representation in order to:

- handle a wider range of inputs (both status and trends)
- predict the effects of effectors stuck in a particular state in future flight phases.

Status values can serve as input because a trend can be calculated from the previous state and the current state. Similarly, at the output, the resulting status can be calculated from the previous state and the current trend following the state transition diagrams shown in Figure 4-7.

Resolution - Selection of Qualitative Values. Qualitative models, by definition, have a lower resolution than the corresponding equations (quantitative models). In constructing a qualitative model, the designer must therefore carefully consider the consequences of selecting a particular set of qualitative values, since all actual values within a given qualitative value are treated identically. It is typically easier to use the sign of the first derivative, where there are natural

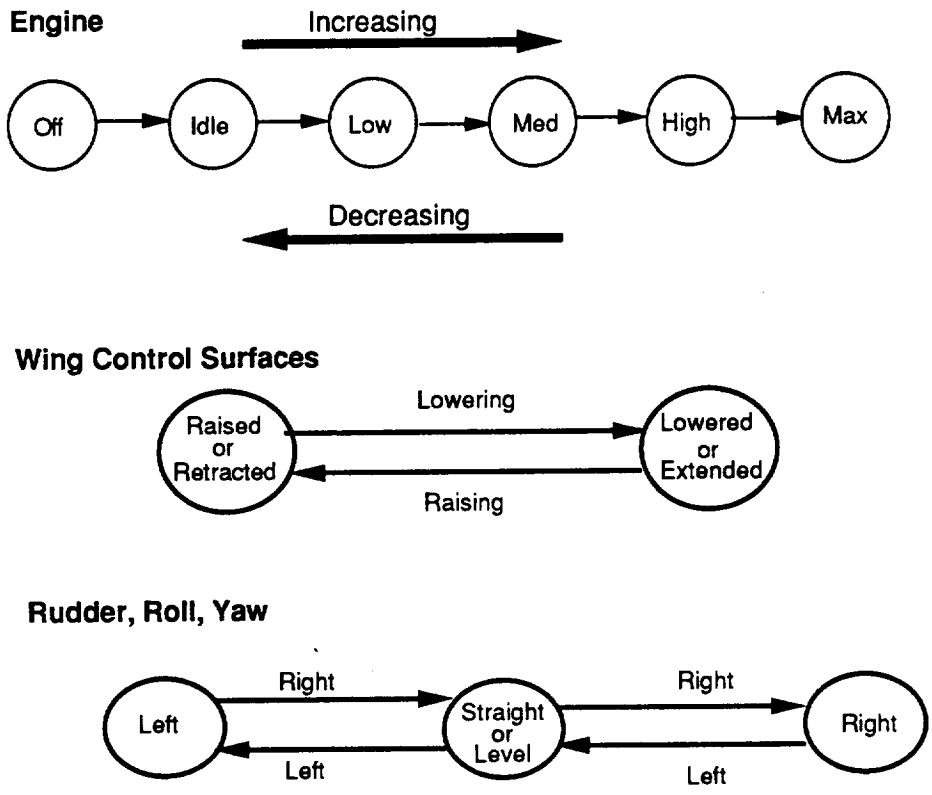


Figure 4-7: State Transition Diagrams Showing the Relationship Between Status and Trend. The States Correspond to the Qualitative State Values and the Links to the Qualitative Trend Values.

breakpoints resulting in three distinct regions: increasing, stable, and decreasing, which are most often represented as -,0, and +. These values correspond to the value of the *trend* attribute. The use of state values is more problematic, since there is a greater number of possible interesting states. For example an engine setting can be off, idle, low, medium, high, or max (expressed relative to the maximal setting). As we will see, a large number of states makes it more difficult to define the arithmetic required for the combination of qualitative values which is necessary any time multiple causal influences exist. According to this criterion, the trend attribute is thus preferred over the status attribute, due to its small number of well defined qualitative values.

Qualitative Arithmetic - Combining Multiple Causal Effects. One of the difficulties with qualitative reasoning stems from the reduced resolution of this representation and the resulting ambiguity of the inferences. This is particularly problematic when causal influences from multiple sources must be combined. Consider the situation where there are three engines contributing to the total thrust. What should the thrust value be if engine 1 is low, engine 2 is idle, and engine 3 is max?

When there are multiple influences (whether of the same type multiple engines affecting thrust; or of different types such as both thrust and pitch affecting airspeed) it becomes difficult to determine what the final qualitative value should be, based on the incoming values. One solution is to increase the resolution of the qualitative states of the object receiving the multiple influences and assign a unique qualitative value to each combination of the incoming values. This is not a satisfactory resolution of the problem, since it leads to a proliferation of somewhat arbitrary qualitative states (really low, somewhat low, medium low, etc.) and a correspondingly awkward arithmetic. This approach is also very domain specific and lacking in generality.

Another problem with qualitative reasoning is the existence of contradictions when opposing causal influences are combined. An example of this is a situation when the pitch is decreasing (leading to an increase in airspeed) but at the same time the thrust is decreasing (leading to a decrease in airspeed). At such points no solution exists at the qualitative level.

The problem we face here is to devise an arithmetic which defines how the qualitative values are combined. The construction of such an arithmetic becomes more difficult as the number of qualitative values increases. It is also made more difficult if the values do not contain a natural zero value (landmark value). A zero value simplifies the arithmetic by reducing the number of value combinations where the operations are undefined. Since the qualitative values describing the status have no zero value, the only time this arithmetic is defined unambiguously is when the multiple values are identical; i.e., low and low give low, high and high give high, etc. Because the set of qualitative values describing the trend does have a zero point (the zero derivative or "stable" trend), the only time this arithmetic is undefined is when the values are opposing; i.e., increasing and decreasing. This follows the established practice in qualitative physics.

Considering the ease of arithmetic definition criterion, the trend attribute is preferred over the status, both due to the small number of qualitative states (typically three), and because of the existence of a zero (any stable state). The arithmetic for one of the sets of qualitative values is shown in Figure 4-8. The problems associated with defining the arithmetic for various quantity spaces are discussed by de Kleer and Brown [1984] and Williams [1984].

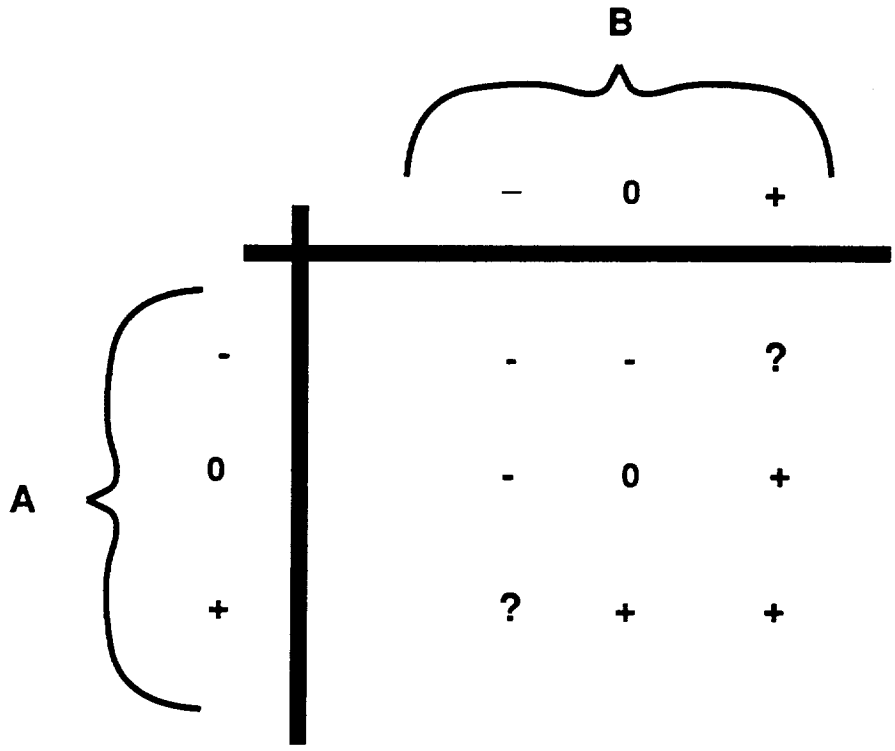


Figure 4-8: Qualitative Arithmetic for Combining Causal Influences (A and B)

Time. Time is not represented explicitly in the current implementation. While the causal representation provides some information about the time relationship among events (i.e., causes precede effects), there are no indications about how long an event will last or when it will occur. However, there is a subtle difference between the use of the status and attributes with respect to time.

The trend attribute represents a change in behavior. Assuming a continued input, a trend in a particular direction will cause a corresponding trend in the causally related objects. That is, if an engine begins to decrease its output, there will be a corresponding decrease in thrust, and eventually a decrease in airspeed. The trend attribute can thus be propagated directly, always

inducing the corresponding change in behavior in the causally related objects. The effects are instantaneous and continue as long as the initial causal influence persists.

This is not the case for the status attribute. Consider the case where the engine output is "low". Does this mean that the thrust will be "low"? Not necessarily, because the new state is a function of the previous state. If the previous state had been "max", then the low engine output will reduce the thrust but not bring it to "low". But this is just the instantaneous effect. Once the system has stabilized, the thrust will be "low".

The trend thus represents an instantaneous change which can be propagated directly. The status represents a more stable aspect of the system and can also be used to propagate causal effects, but is only valid once the system has stabilized. Since the faults presented to the FaultFinder, and thus to RECORS, typically represent sudden changes in behavior, we have decided to use the trend attribute as the primary value to propagate in order to simulate fault behavior.

4.2.4 Model Links (Model Object Relationships)

4.2.4.1 Causal Links

The model contains several types of causal links as well as taxonomic links combined into a single structure. These are discussed below.

Single Object Links. Objects that influence one another are connected by causal links. There are two causal links for each abstraction level: forward *(affects)* and backward *(affected-by)*. At the binary level, these links simply list the objects affecting or affected by the current object. For example, the *affected-by* attribute of the object *elevators* points to the object *lift-tail* which in turns points to *pitch*. At the qualitative level, the causal links contain constraint expressions describing the relationships among the affected objects. The syntax of these expressions is: (<current trend value> <affected/affecting-object> <affected/affecting object's value>). For example, *thrust-wing-left* is proportional to the output of the *engine-wing-left*. Thus, the *affected-by* attribute of *thrust-wing-left* contains the following expressions: (increasing *engine-wing-left* increasing) (decreasing *engine-wing-left* decreasing). Similarly, the affects

attribute of *engine-wing-left* contains: (increasing *thrust-wing-left* increasing) (decreasing *thrust-wing-left* decreasing). A more efficient representation [Rajagopalan, 1984] would simply indicate that the relationship is directly or inversely proportional rather than explicitly listing the values. However, in terms of their inferential capabilities, these two representations are identical.

Causal Links Involving Relations Among Multiple Objects. The core causal model (Figure 4-3) represents causal links among individual objects. It does not necessarily represent the causality involved in a particular type of relationship among several objects. This type of causality occurs any time two effectors are intended to function in a symmetrical mode. In such cases, asymmetry will create new causal effects. For example, the particular type of relationship among the thrusts on the wing engines (equal or asymmetrical) will itself have a unique causal link to a flight characteristic: yaw.

In cases where multiple effects exist, one will often dominate in a given situation. For example, the lift on the two wings affects both the roll and the altitude.[7] However, if this lift is asymmetrical, the roll effect will be primary, and we would not want to infer that increasing the lift on one of the wings will affect the altitude. However, if the lift on both wings is altered equally, the primary result will be a change in altitude (e.g., when spoilers are used for quick descent.) The effects of these types of relationships are represented in a separate part of the knowledge base, the *Multiple-object-constraints* (see Figure 4-9).

These relationships may represent both normal (equal lift on wings affects altitude) and abnormal (asymmetrical lift affects the roll) situations. The causal links from these objects to the objects representing the flight characteristics represent how these relationships influence the flight characteristics. The names of all constraints associated with a particular object are contained in the attribute *constraints*. This attribute is checked during simulation and the individual constraints are checked to see if they are satisfied. For example, the constraints attribute of lift-wing-left points to the object *asymmetrical-lift* in the knowledge base

[7]The exact effect on the altitude is expressed by the relationships between the status and trend attributes of the objects involved.

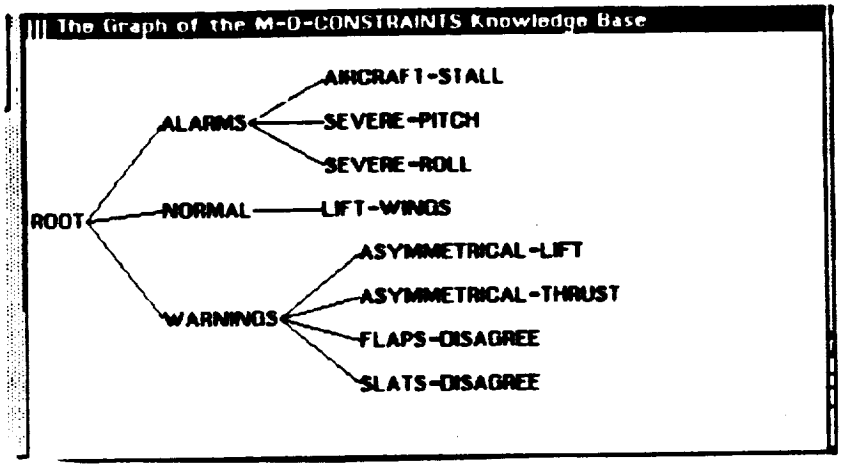


Figure 4-9: The Knowledge Base Containing the Objects Representing Constraints Among Multiple Objects in the Core Model.

Multiple-Object-Constraints. Figure 4-10 illustrates these types of relationships and the connections between the two knowledge bases.

4.2.4.2 Taxonomic Links - Taxonomic Hierarchies

In addition to the causal links, the model also contains taxonomic links, which relate objects by the class membership relation. The taxonomic links give rise to the taxonomies shown in Figure 4-5. This hierarchical organization allows both an efficient representation of objects, by allowing the inheritance of attributes and their values, and provides additional levels of abstraction (orthogonal to the binary and the qualitative levels) which support reasoning in the absence of more detailed information about the state of the aircraft.

Representational Efficiency. This is achieved by grouping together objects which exert similar causal effects and representing the causal links only once. For example, all control surfaces on the wings affect the lift in similar ways: raising the control surfaces (spoilers, ailerons, flaps, slats) results in a decreased lift; lowering the *surface* results in a increased lift.

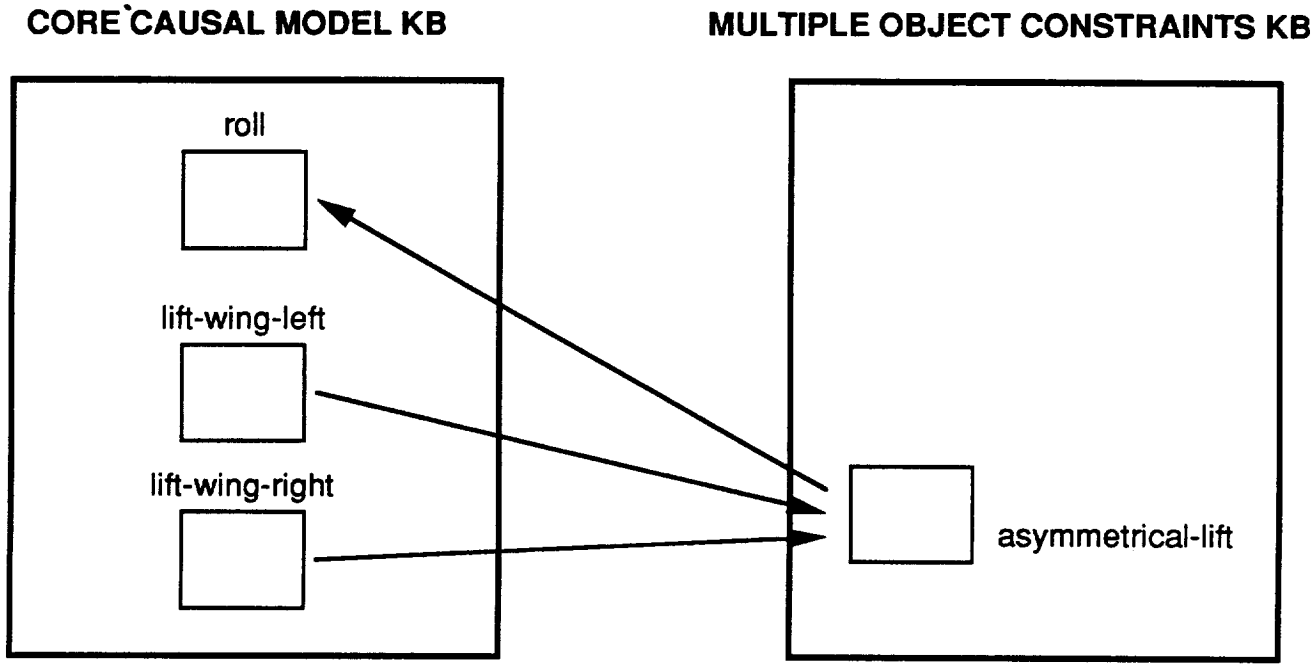


Figure 4-10: Relationship Between the Core Causal Model and the Multiple-Object-Constraints Knowledge Base.

This relationship is represented just once, at the level of *control-surfaces-wing*, rather than for the individual control surfaces. Similarly, the contribution of the individual engines to the total thrust and thus the airspeed is represented in one place: the thrust to airspeed causal link.

Taxonomic abstraction levels. These levels of abstraction correspond to the different levels of the taxonomy. They are similar to the binary and the qualitative levels in that the higher levels can be used to make inferences about the aircraft state in the absence of more exact data. For example, by explicitly representing the three engines as a separate object *engine* in the model, and connecting this object to the causally related objects *(thrust and airspeed)*, the model can support inferences given only the information that "something is wrong with the engines". A detailed description of reasoning at the different levels of taxonomic abstraction is in Section 4.3.2.

4.2.5 Combining Causal and Taxonomic Links in the Model

While the combination of the taxonomic and causal representations provides advantages (a uniform structure), it also creates problems. Constructing a model is more difficult than putting together an set of unrelated rules. Some of these issues are discussed below.

Separating Causal Relationships. Causal effects common to more than one object are represented at those objects' parents. This occurs both in the case of the individual control surfaces, whose effect on lift is expressed once at the level of *control-surface-wing-left*, (Figure 4-11a) and in the case of thrusts from the individual engines, whose effects on speed are expressed once, at the level of the parent object, *thrust*. We have to be careful to make sure that an object does not combine inappropriate causal effects. It is tempting to create such objects, since the physical grouping often suggests a corresponding model grouping. For example, it is tempting to group the control surfaces on the tail together, much like we did the control surfaces on the wing. The resulting model is shown in Figure 4-11b. However, this would be a mistake, since the tail control surfaces, rudder and elevators, affect two different forces: *f-yaw*[8] and *lift-tail* and subsequently the *yaw* and the *pitch*. The model that illustrates the correct representation is shown in Figure 4-11c. The rule to follow here is this: An object should represent a causal link to another object only if its children can inherit this causal link; in other words, if all its children are similarly related to the other object.

Inheritance of Causal Relationships. Inheritance, however, presents some difficulties. In principle, any object can inherit all of its parents' causal links. However, since these links are expressed at higher levels of abstraction, it would not necessarily be appropriate to inherit them. For example, the *hydraulic-systems* object is causally related to the *control-surface* object. If these causal links were inherited, then the *affects* attribute of *hydraulic-system-A* would contain all affected effectors plus the object *control-surface* which would lead to redundant inferences.

When inheriting causal links we must therefore make sure that only the most specific links are used in inferencing.

---

[8] *F-yaw* represents the side force induced by the rudders.

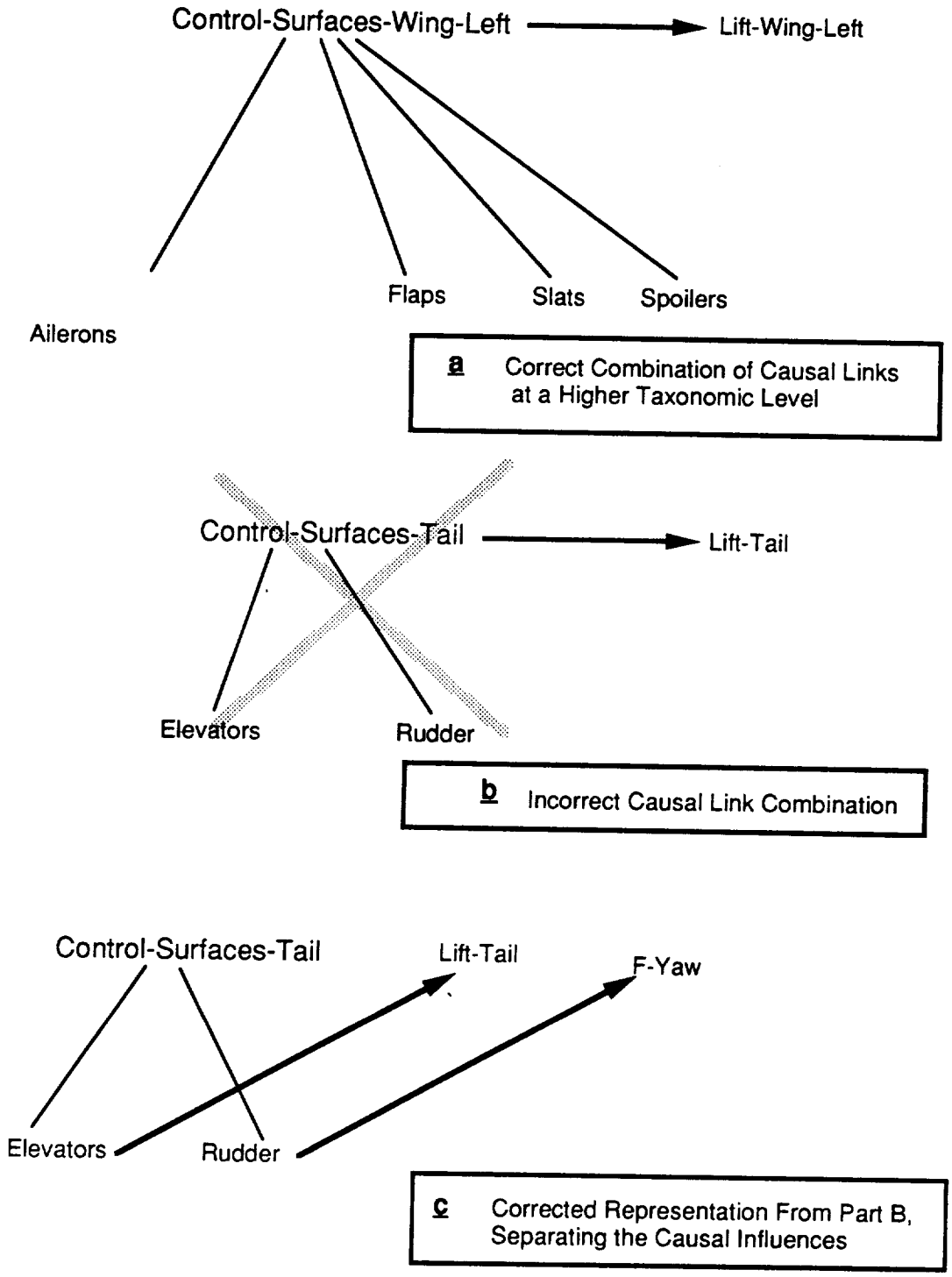


Figure 4-11: Correct and Incorrect Representation of Causal Pathways.

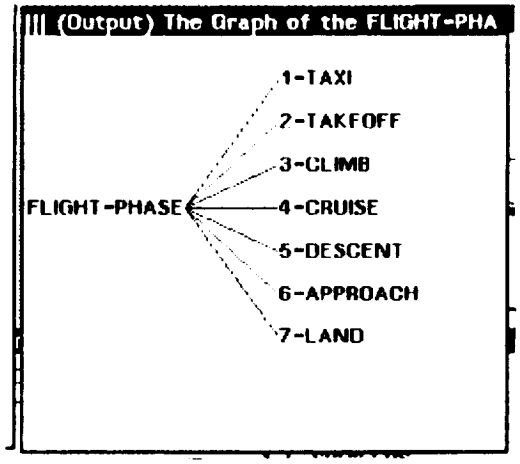
Link Types. Although the taxonomy, by definition, is structured according to class-subclass (also referred to as taxonomic or IS-A links) links, it also captures some other relationships, such as is-located-on (individual control surfaces are located on the left wing, represented by *control-surf-wing-left*), is-part-of (lift of individual wings is part of total lift; thrust of individual engines is part of total thrust). We take advantage of using these links for the types of inferences they naturally support. For example, if the left wing has been damaged, the model can use the *control-surf-wing-left* object and its causal links to infer that the lift on the left wing will be affected and a roll may result. The model also supports the inferences made by following the taxonomic links to the children, and deducing that any of the control surfaces on that wing may be damaged. The causal links from these objects can also be followed, and the full range of the possible effects of wing damage may thus be deduced.

4.2.6 Representing Desired States and Behaviors

So far we have not discussed any mechanism for determining whether the behavior or status of the system is desirable. At the binary level, the implication is that the abnormal value is not desirable. In order to determine whether a particular status or trend is appropriate at the qualitative level RECORS must have a representation of the expected values for each phase of flight. This knowledge can then be used to determine what the corrective response should be in an undesirable situation.

In RECORS, the desired aircraft status and trend values are expressed in constraints attached to the model objects' attributes. We distinguish between several types of constraints: those that must hold at any point during the flight (alarms and warnings) and those that are specific for a given flight phase and thus change with time (flight phases). Examples of the former are asymmetrical-thrust and aircraft-stall. Examples of the latter are the different effector settings and flight profile characteristics required in the different flight phases. Figure 4-12 shows the knowledge base representing the different flight phases as well as the internal structure of the objects.

ORIGINAL PAGE IS
OF POOR QUALITY



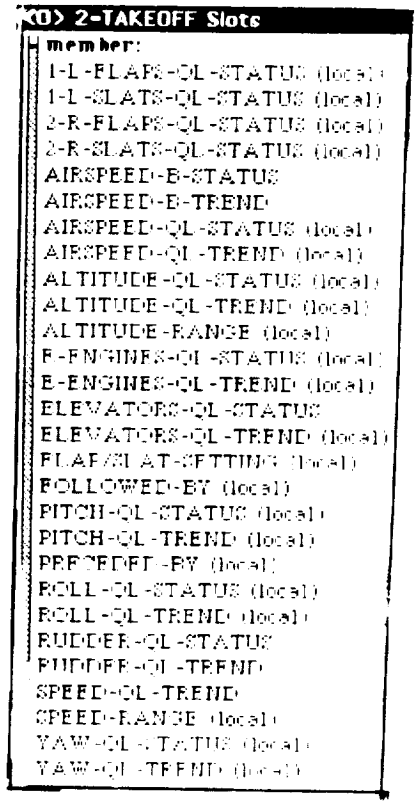


Figure 4-12: The Knowledge Base and Objects Representing the Flight Phase Constraints.

4.2.7 Alarms and Warnings

Both alarms and warnings represent undesirable situations which should be avoided. The difference between the two is that some of the states which trigger a warning could in fact be desirable in abnormal situations and could be used to achieve a desired response in an emergency. For example, if the rudder failed, asymmetrical thrust could be used to induce a yaw. Similarly, if the ailerons failed, asymmetrical lift induced by an asymmetrical setting of one of the other wing control surfaces could induce a roll.

Unlike the warnings, alarms represent situations which are always undesirable. For example, while an aircraft stall will result in a decreased altitude, it is not generally a good idea to stall the aircraft in order to descend faster.

Each alarm and warning exists at both the binary and the qualitative levels. When an alarm or warning is triggered at the binary level, the status of the corresponding object becomes abnormal. At the qualitative level, the trend of the object takes on the corresponding qualitative value. For example, in the case of asymmetrical lift, the value will be either left or right.

4.2.8 Flight Phase Constraints

The flight phase knowledge base lists the desired effector settings and flight characteristics for each of the seven flight phases. This information is used during simulation to determine which flight phase goals, if any, are violated by a fault. The violated goals are displayed to the pilot, in addition to the triggered alarms and warnings and the affected flight characteristics. During response derivation the flight phase constraints are used to determine whether a suggested response is feasible; i.e., whether it is consistent with the current flight phase constraints.

4.3 Inferencing Mechanisms

RECORS's knowledge bases support several types of reasoning, including the simulation of identified faults and the generation of possible responses to achieve a desired set of flight characteristics. The underlying mechanism used to implement these is bi-directional value propagation through the causal links at either the binary or the qualitative level, at various levels of the taxonomic hierarchy.[9] During simulation, the initial value is provided by the FaultFinder and represents a faulty subsystem or aircraft effector. Simulation output consists of affected flight characteristics as well as any violated flight phase goals, alarms, and warnings. During response derivation, the initial value represents some desired flight characteristic. The output consists of effector settings which would achieve the desired state, taking into consideration the current state of the aircraft (e.g., failed engines) and current flight phase goals (e.g., maintain increasing altitude).

Simulation and response derivation are linked by the goal generation process, which maps the warnings, alarms, and violated flight phase constraints into desired flight characteristics that would help correct the current state. Figure 4-13 shows how these types of reasoning are related and how they use RECORS's causal model. Figure 4-14 illustrates in more detail the simulation algorithm which is discussed in Section 4.3.3.

4.3.1 Multiple Failure Simulation/Multiple Response Derivation

If we view only the causal graph portion of the model we can see several places where links either converge or branch out (see Figure 4-15). These points indicate interacting multiple failure paths or multiple response possibilities. Simulation of multiple failures is achieved by sequentially propagating the corresponding values through the model. If the failures interact, their propagation paths meet, and the corresponding values are combined according to either the binary or the qualitative arithmetic described in Section 4.2.3. Depending on what the values are, this may lead to a contradiction. During response derivation, reaching these points means

[9]This mechanism can also be extended to include diagnosis.

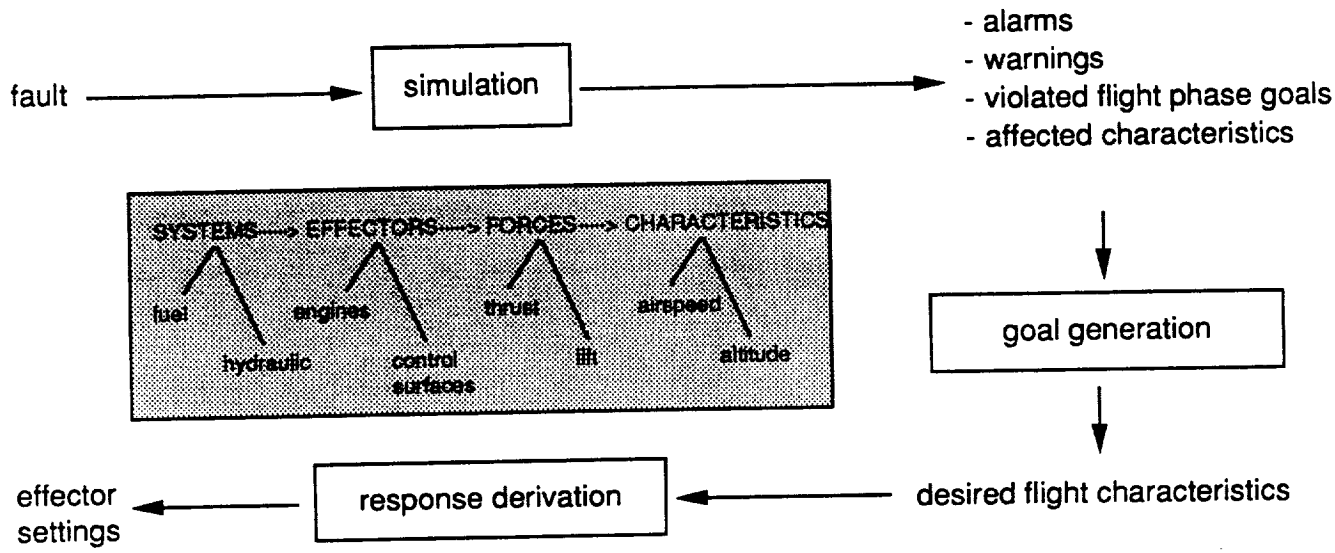


Figure 4-13: A High-Level View of the RECORS's Reasoning Types.

that there are multiple ways of achieving a certain state. In such cases there is typically a preferred order in which these responses should be attempted; e.g., if the ailerons can be used to induce a roll then an asymmetrical spoiler setting should not be used. This issue of preferred causal paths does not arise during simulation, because any possible causal propagation pathway must be followed to make the full set of inferences required to represent the effects of a fault.

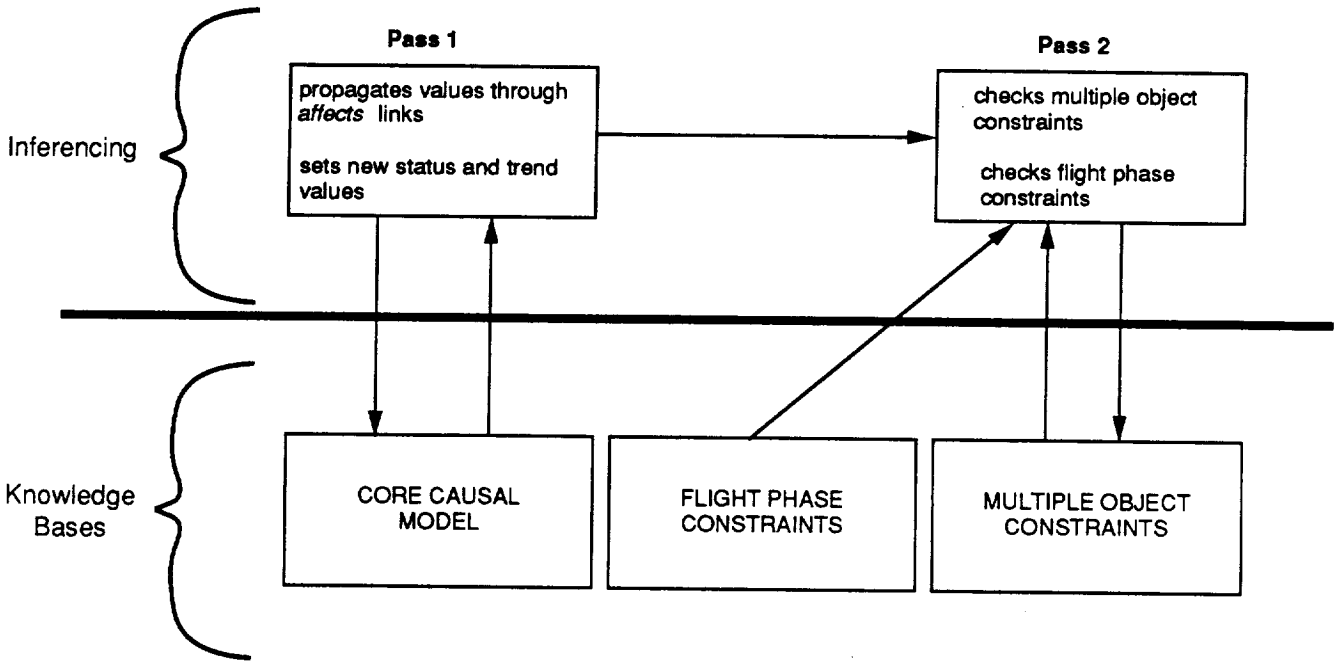


Figure 4-14: Simulation Process Diagram

4.3.2 Reasoning at Multiple Levels of Abstraction

Section 4.2 described the different types of representational abstractions that exist in RECORS's knowledge bases; binary and qualitative on the one hand, taxonomic on the other. Reasoning can take place at any of the these levels and transitions among the levels are possible. In cases where little information about the aircraft state or fault is available, the higher abstraction levels allow correspondingly more general inferences. If more detailed results are

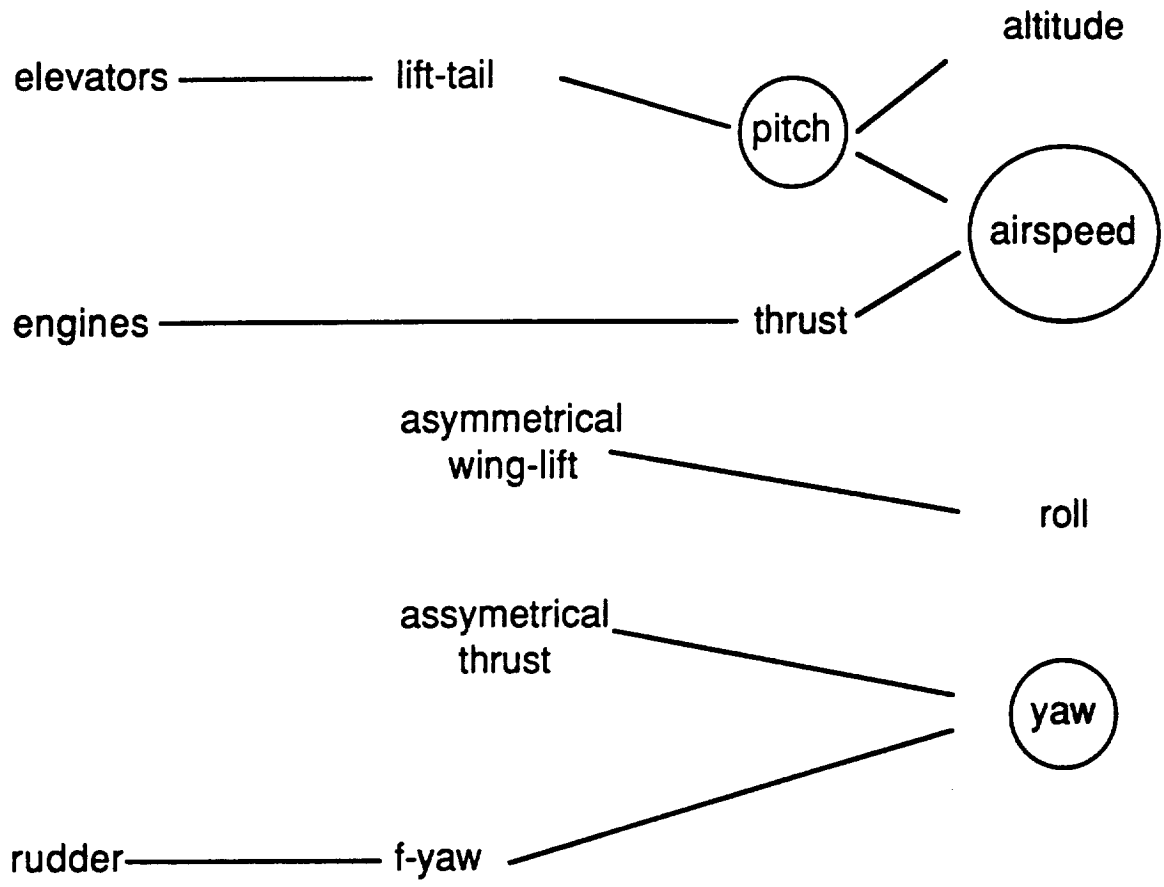


Figure 4-15: The Causal Graph Defined by the *affects* and *affected-by* links.

required, the reasoning can switch to a lower level of abstraction (from binary to qualitative, or from a higher taxonomic level to a lower one) and explore the full space of possibilities at that level.

There is, however, a tradeoff between the level of abstraction and the certainty that the inferences are accurate. If the initial data exists at a higher level of abstraction, we can make inferences at that level and maintain the certainty that these inferences are correct. When we switch to a lower level, the certainty is reduced, since we are now exploring the space of possible results. For example, if we know that "something is wrong with the engines" we can infer that "something will be wrong with the thrust". However, if we switch to a lower level of

abstraction and infer either that "the thrust is low" "thrust is high" or that "thrust left is abnormal" or "thrust right is abnormal" then the certainty of these inferences is reduced.

The type of reasoning used initially depends on the input data. Figure 4-16 shows examples of input at the different levels of abstraction.

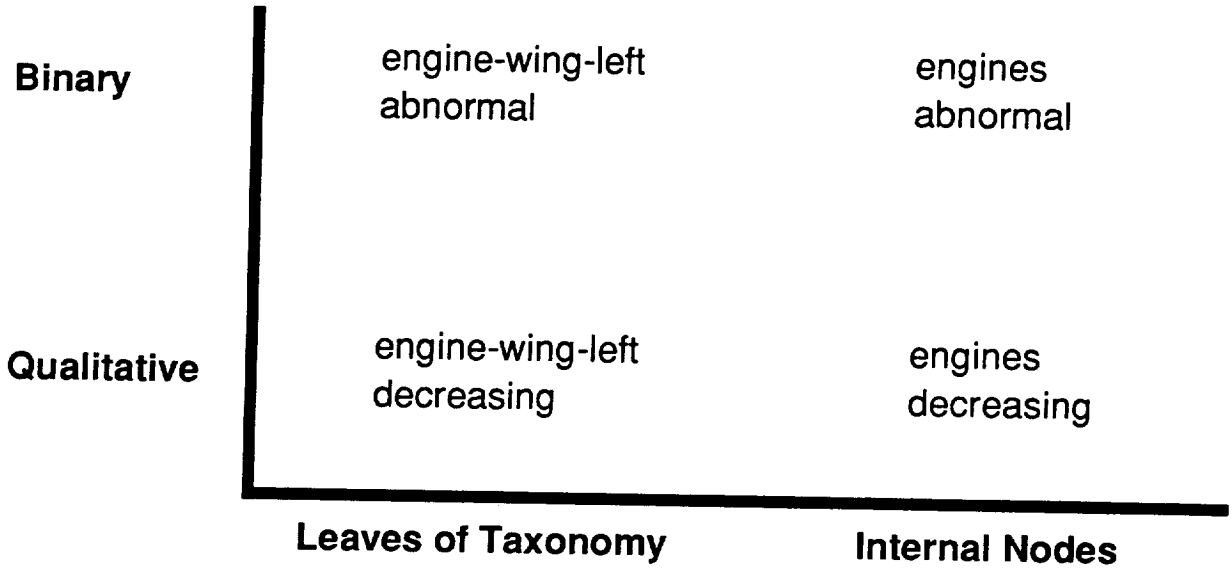


Figure 4-16: Examples of Input Data at Different Levels of Abstraction.

How the reasoning progresses depends on both the results of the process (whether contradictions are reached), the initial level, and on user-defined parameters which control transitions among the different levels of abstraction. These different transitions are described below.

Qualitative to binary. This transition is useful when a contradiction is reached at the qualitative level. Contradiction is reached any time opposing causal influences must be combined, resulting in an undefined value. This occurs primarily when multiple failures are simulated, for example, if the the thrust of one engine is decreasing while the other is increasing. In such cases the reasoning would continue at the binary level and conclude that airspeed will be abnormal, without specifying whether it will decrease or increase.

Binary to qualitative. Whether binary reasoning is taking place due to the initial data or because of a previous transition due to a contradiction one way to obtain more detailed inferences is to make a transition to the qualitative level and explore the full space of possibilities at this level. This full search is feasible because there are only three qualitative states. For the example above, this would mean simulating both an increased thrust and a decreased thrust. The advantage of this approach is that any alarms or warnings associated with the qualitative values will be triggered and any causal links represented at the lower levels will be activated.

Switching to a lower taxonomic level. This is accomplished by expanding model objects at the higher levels of the hierarchy into their constituent objects (following the taxonomic links) and presenting this information to the crew. For example, if the input specified that "something was wrong with the engines" we could expand the "engines" object to the individual engines and inform the crew that there is a chance that any of these engines are faulty. This is feasible as long as the number of objects at the lower levels remains small. In the current version of the model this number is typically between three and five (see Figure 4-5). In this case however the inferences about these objects have a lower belief, since we do not know exactly which of the engines or control surfaces is malfunctioning. For example, if the thrust is abnormal, we can expand the thrust object into the three constituent components and present this information to the crew.

The other advantage is that the causal links at the lower levels are activated and can both provide more information (e.g., if a wing engine has failed, then the yaw will be affected) and help identify the actual error (the direction of the yaw indicates which engine is affected).

Constraining the Answers Generated at the Lower Abstraction Levels. Whenever a transition is made to a lower level of abstraction, whether from binary to qualitative or from a higher taxonomic level to a lower one, there is the problem of many possible answers. This number can be reduced by eliminating the answers whose results are not consistent with the current flight characteristics. For example, if a taxonomic level expansion suggests that either the left wing, right wing, or tail engine have a decreased output and at the same time the plane is yawing to the left, we can deduce that it is in fact the left wing engine whose output is low.

In the future, other sources of information could also be used to increase or decrease the belief in the particular inference. For example, the maintenance record may show that the left wing engine tends to have problems and this can help increase the belief that it is this engine which is faulty.

Controlling the Transitions. In the current version of the system the user specifies which of these possibilities is desirable; the default is to change to the binary level when a contradiction is reached at the qualitative level. The user can also control these transitions by an explicit request through the interface or by modifying the system parameters.

4.3.3 Simulation - Forward Value Propagation

At the most general level of description, simulation consists of propagating a status or trend value of some object forward through the model's causal links. The input consists of data from the FaultFinder and can be either at the binary or the qualitative level, representing either the status or trend of either the subsystems or the effectors. Examples of the former are: flaps extended, elevators up, engine low; examples of the corresponding trends are: flaps extending, elevators raising, engine decreasing. In addition to these inputs, which represent changes in the current state of the aircraft, we assume that the model has available the current state of the aircraft subsystems and effectors, which is automatically, updated as these states change. We assume that any discrepancies are detected by the FaultFinder and thus RECORS processing (fault simulation) is triggered by the FaultFinder's input. Figure 4-17 illustrates the relationship between RECORS and the FaultFinder.

The simulation works in two passes. In pass one, the values are propagated through the core model, resulting in the status or trend attribute values of the model objects being changed to reflect the effects of the simulated faults. In pass two[10] the constraints representing alarms and warnings are checked, as well as any multi-object causal relationships. Any violated flight phase goals are collected during both passes and are displayed to the crew. These violated goals, as

[10]The reason for the two passes is that the multi-object constraints may not have all the necessary information available during pass one.

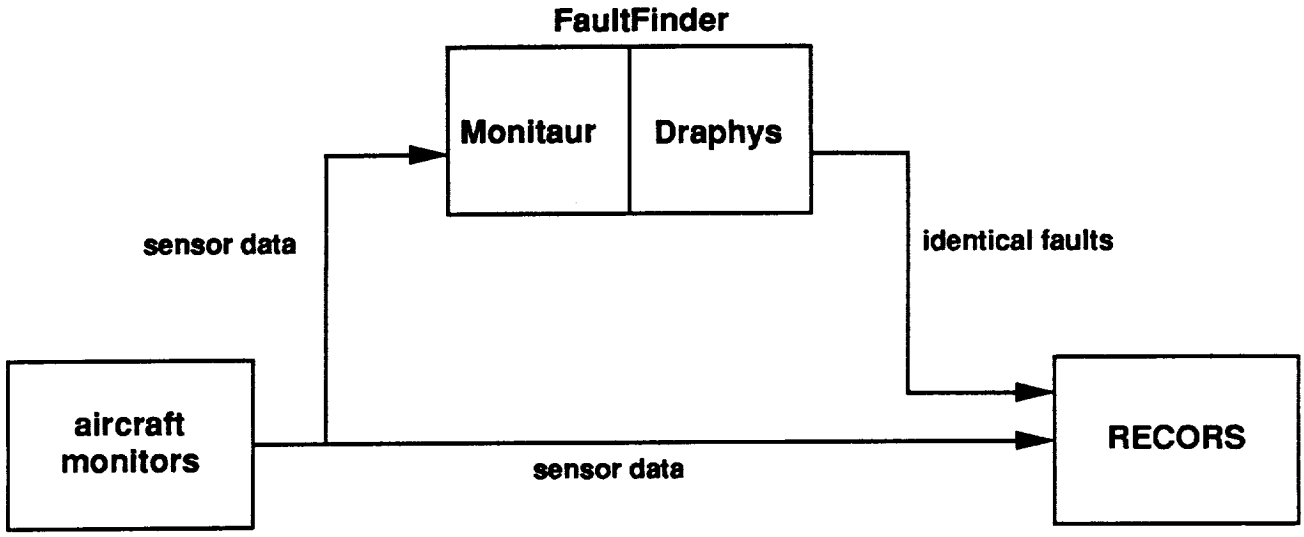


Figure 4-17: Relationship Between RECORS and the FaultFinder System

well as any triggered alarms and warnings, comprise the input to the goal generation procedure, which is described in Section 4.3.4. Figure 4-14 shows a diagram of this process. The following example illustrates this process.

Suppose RECORS receives as input the information that left wing engine's RPM's are decreasing. This qualitative trend value is propagated through all causally released objects in pass one. This causes the ql-trend value of the objects thrust-wing-left and airspeed to be changed to "decreasing". In pass two any multiple object constraints are checked. In this particular causal chain there is only one such object asymmetrical thrust, which represent the relationship between the wing engine thrusts. The asymmetrical thrust is causally related to yaw, therefore the ql-trend value (left) is propagated forward, leading to changes in the ql-trend value of the yaw object. At this point the goals associated with the particular flight phase are checked, and a violated goal is found in both the airspeed and yaw. These results are displayed to the crew.

4.3.3.1 Using the Taxonomic Links of the Model

The taxonomic organization of the model makes it possible to specify similar causal relations among several object at one point, higher in the taxonomic hierarchy. Thus the fact that thrust for all three engines affects airspeed is only represented once, in the *affects* attribute of the object *thrust*, which represent the total thrust. An analogous relationship exists between the contributions of the individual control surfaces to the overall wing lift. Thus not all causal relationships are represented at the same level of the taxonomic hierarchy. The simulation algorithm must therefore use both the affects links (to access causal relationships at the same taxonomic level) and follow the taxonomic links up the taxonomic hierarchy, to access any causal relations that may exist in the parent objects at the next level.

4.3.3.2 Binary Simulation

Binary simulation is used when only non-specific information is available about the state of the system. It is triggered whenever RECORS receives abnormal status or trend of some subsystem or effector. (Recall from Section 4.2 that at this level both trend and status contain same type of information and are thus treated identically.) The abnormal value is propagated via the "affects" links of each object. These links contain a list of the causally related objects. The simulation algorithm first looks up the *affected* objects in the affects attribute and then sets the status to abnormal. This process continues until objects are reached which have an empty "affects" attribute, indicating no further causal connections. In the current version of the model these are the flight characteristics. The output consists of all abnormal flight characteristics, triggered alarms and warnings, and violated flight phase constraints. Note that at this level there is no further information about the way in which the characteristic has been affected. All we know is that, for example, the thrust is abnormal, but not whether it is low or high. Examples of output at this level are: airspeed or altitude is abnormal, asymmetrical-lift exists (but not the direction), roll or yaw are abnormal (but not the direction).

4.3.3.3 Qualitative Simulation

Qualitative simulation is used when more precise information is available from the FaultFinder describing the state or behavior of affected system or component in terms of one of the possible qualitative values. (Recall from Section 4.2 that only trend values are propagated during qualitative simulation.) If a status value is received, it is first translated to trend by comparing the new state with the existing state following the diagram in Figure 4-7.

Current \ Desired	-	0	+
-	no Δ	+	+
0	-	no Δ	+
+	-	-	no Δ

Figure 4-18: A Table Relating Current Trend Values to Desired Trend Values

The simulation algorithm first selects the appropriate constraint expression from the list of such expressions in the *ql-affects* attribute of the current object. This selection is done by matching the current trend value with the first value of each constraint expression. For example, if we wanted to simulate the effects of a decreasing thrust, we would follow these steps:

- Access the *ql-affects* attribute of *thrust* to obtain the following set of constraint expressions: (increasing airspeed increasing)(decreasing airspeed decreasing)
- Match the current trend value of *thrust* with the first values of these expressions to obtain: (decreasing airspeed decreasing)
- Go to the object named in that expression: airspeed

- Set the trend attribute of that object to the last value of that expression: decreasing

This process continues until the a *ql-affects* link is reached which is empty. The output consists of the same set of objects as the binary simulation output, but this time we know the objects' qualitative values.

4.3.4 Determining Desired Corrective State

After the effects of a fault have been determined, the goal generation mechanism determines desired change in flight characteristics that would counteract the problem. The input to this process consists of the violated flight phase constraints and the output consists of the desired set of flight characteristics which would bring the system back to the desired state. Since these flight characteristics are the goals to be achieved, we term this process goal generation. The goal generation consists of mapping the deviation from the desired state or trend of the flight characteristics into an action that will bring the aircraft to the desired state, as represented by the flight characteristics. Since we are currently not considering multi-step sequences of actions, this process is simple.

Binary. At the binary level the goal generation simply provides the violated flight phase constraints to the response generation algorithm, which then lists the effectors that could be used to alter those flight characteristics.

Qualitative. At the qualitative level, the desired flight characteristic can be expressed as either status or trend. If it is expressed as a trend, the goal generation uses tables such as the one shown in Figure 4-18 to determine, based on the current and desired trends, what the change in trend should be. This is then provided to the response generation algorithm. If the desired characteristic is expressed as a status, then the state diagrams shown in Figure 4-7 are used to determine what change in the current trend would achieve the desired state. For example, if the aircraft is yawing to the left, the corrective action is to yaw to the right.

4.3.5 Response Derivation - Backward Value Propagation

The response derivation mechanism receives input from the goal generation in the form of the desired flight characteristic status or trend. It propagates these desired values backward through the *affected-by* links to determine which effectors can achieve this goal, taking into consideration any non-functional components and the current flight phase constraints. In terms of the link traversal in the causal model, the response derivation algorithm performs the inverse of the simulation algorithms.

At the binary level, the output consists of the list of effectors since the resolution of this level does not permit the derivation of the effector settings. The assumption being that since the pilot is already partially aware of the situation simply suggesting the use of a particular effector may be sufficient to trigger the correct effector setting. At the qualitative level, the output provides the suggested effector setting as well, for example, decreasing the thrust or raising elevators. Response derivation works in two passes. First pass consists of propagating the desired characteristic backward and determining which effectors should be used to achieve that value. This takes into account any non-functional effectors, whether it be because the effectors themselves are faulty or because their underlying subsystems are. In the second pass these suggested effector settings are propagated forward, using the simulation algorithm, to see whether their use would lead to undesirable effects (trigger alarms, warnings, or violate current flight phase goals). The effector settings which do not violate any flight phase constraints are then displayed to the flight crew.

4.4 Interfaces

This section describes the interface design of RECORS and user interaction with the system. In designing RECORS interfaces we have provided:

- direct manipulation interfaces where possible,
- displays or icons familiar to the pilots, and
- the same display for both input and output where possible (distrol displays).

Some of these displays are used for developmental purposes and system control (see Figures 4-19 and 4-20); others could eventually be incorporated into the cockpit displays (see Figure 4-21). The design presented here is still preliminary and we expect to conduct a number of experiments in the next phase of the project to evaluate and modify this design.

4.4.1 Instrument Panel: Flight Characteristics

The instrument panel is shown in the "characteristics display" portion of Figure 4-21 and shows the displays that are always visible to the pilot. These are: airspeed, altimeter, artificial horizon[11], and the rate of climb indicators. This set of displays corresponds to the flight characteristics portion of the model.

Input. The instrument panel is used to provide the input to the response derivation algorithm in the form of desired flight characteristics to be achieved by the aircraft. To input a value the user must first choose between *binary* and *qualitative* level from a menu. When *binary* has been selected, the user can click on any of the displays on the instrument panel to select the appropriate flight characteristic. When *qualitative* has been selected, the user must further specify whether a status or a trend value will be input. Each display is then overlaid with a display which divides the set of values into the appropriate qualitative regions, which are mouse-sensitive. When one of these regions is selected, the selected value is assigned to the corresponding attribute of that flight characteristic.

[11]The artificial horizon combines the pitch and roll characteristics and include a yaw indicator in the bottom portion of the display.

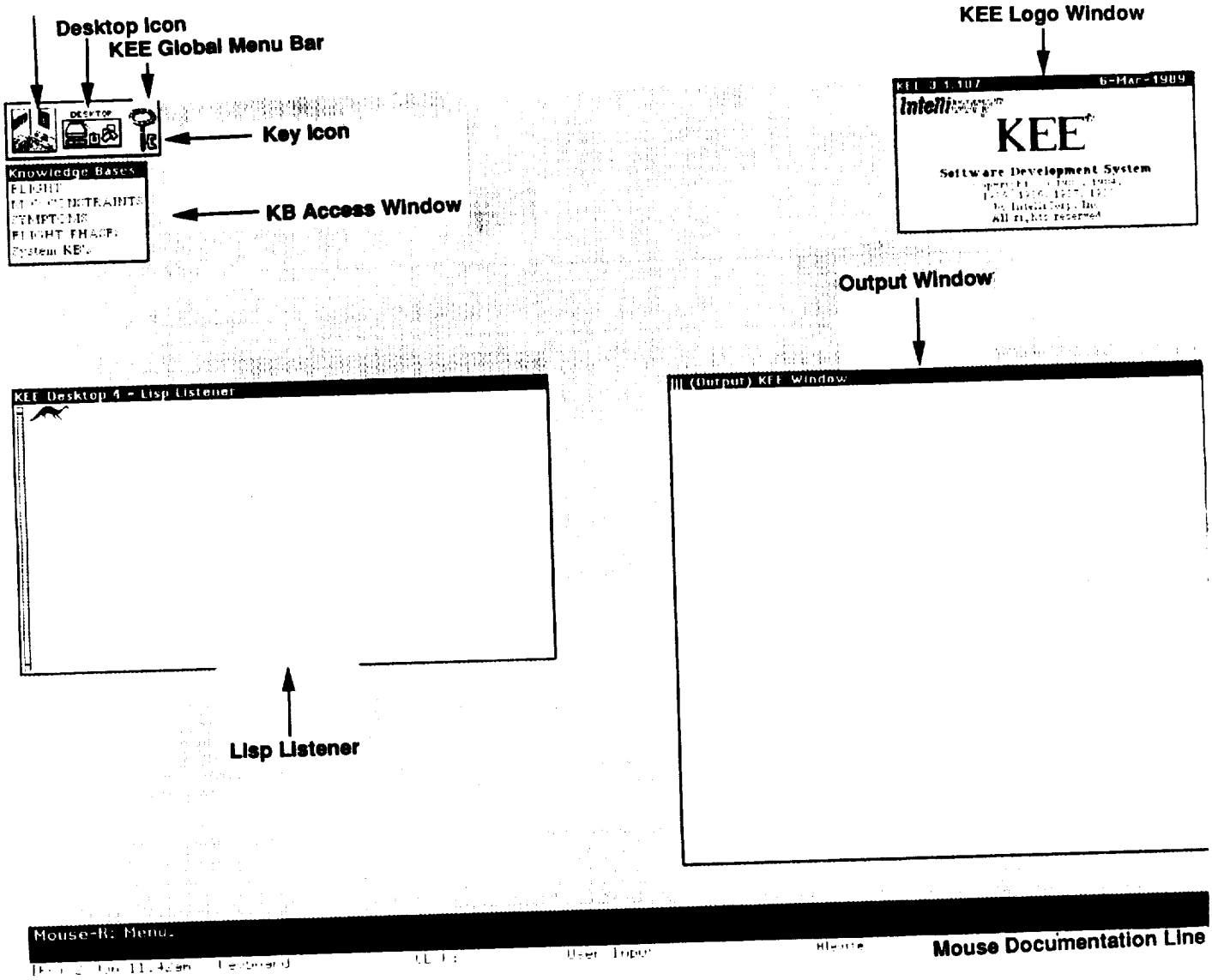


Figure 4-19: KEE Development Screen

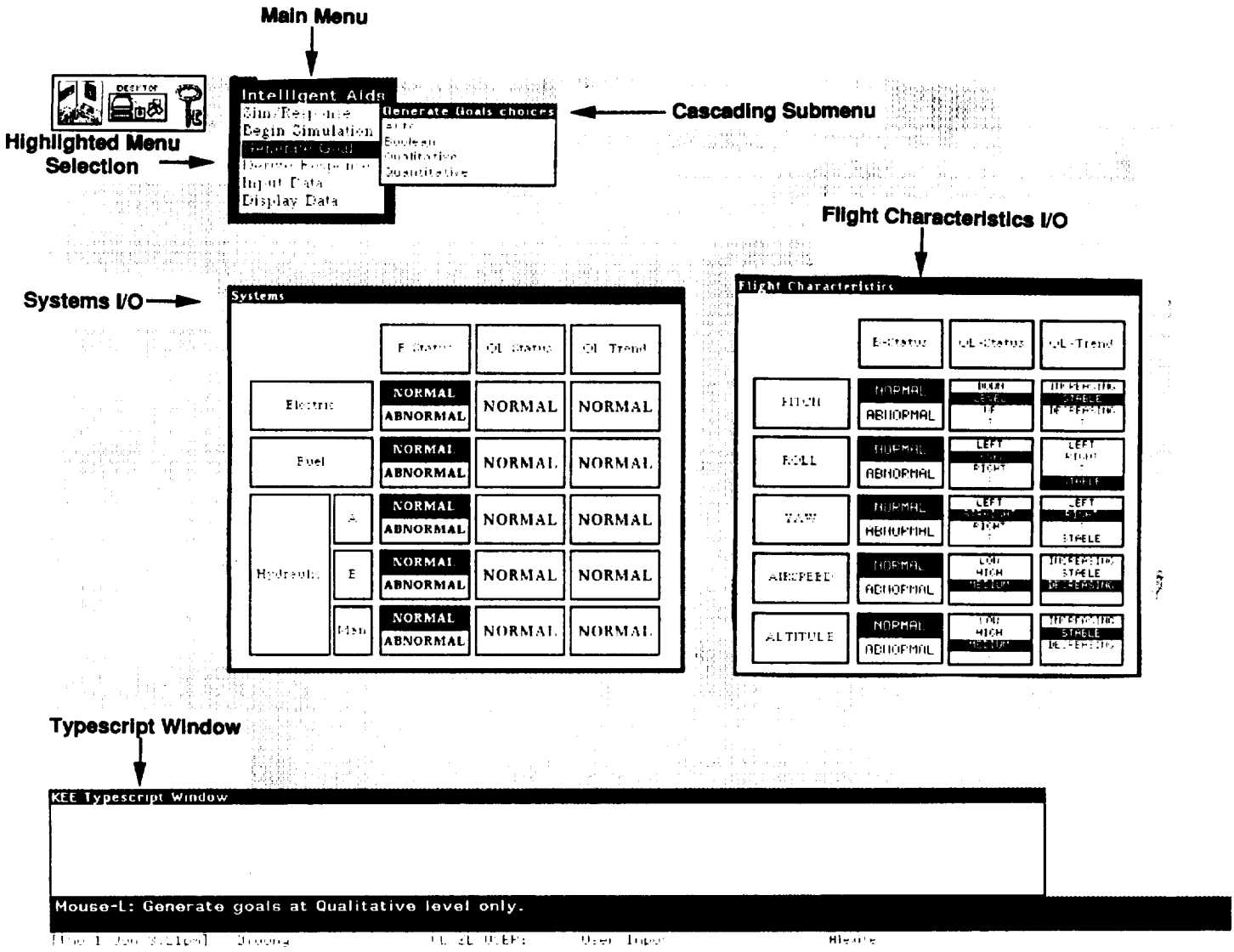


Figure 4-20: RECORS Control Screen

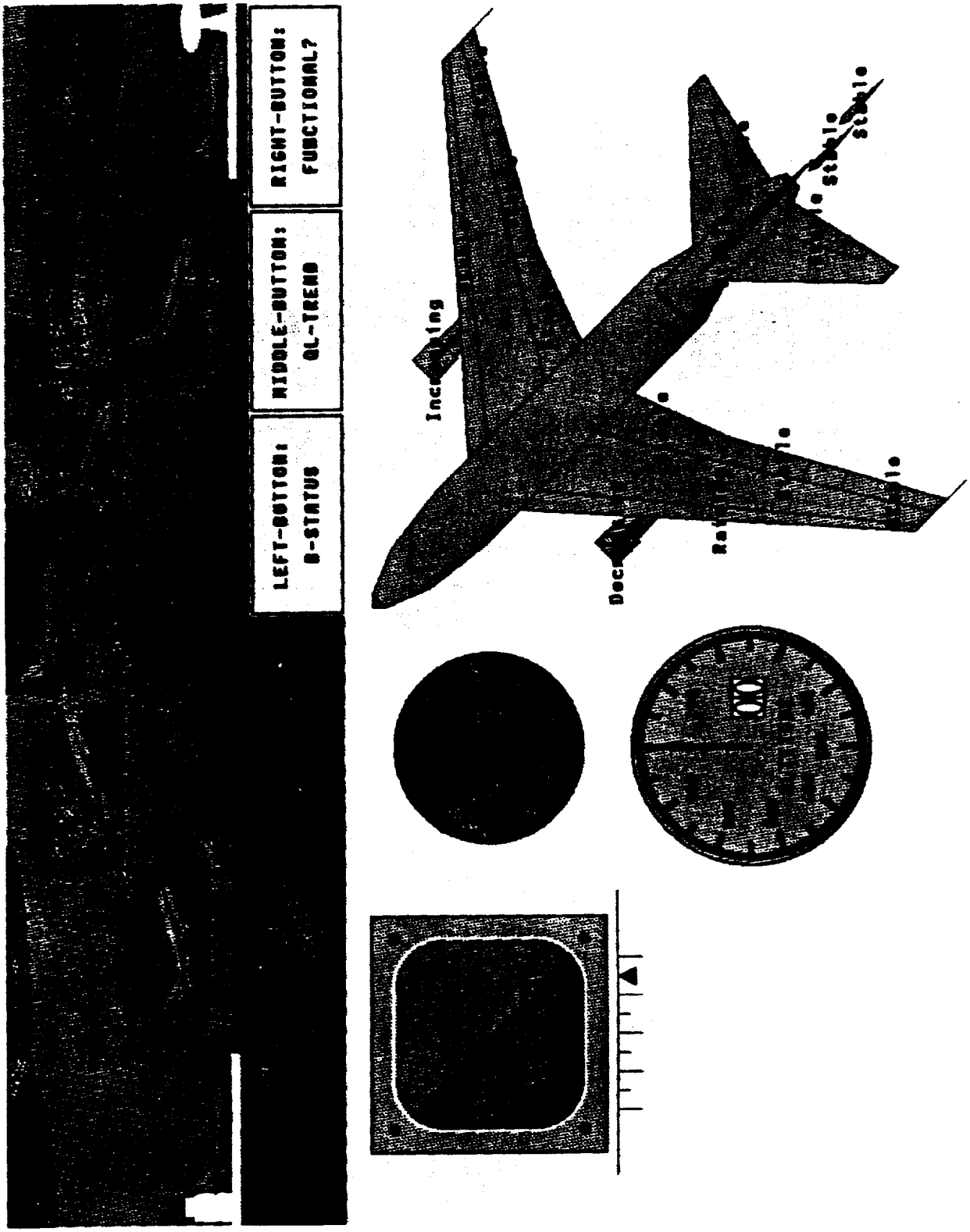


Figure 4-21: Color DIsplay Screen

Output. The output is used to display the flight characteristics affected by a fault. These are displayed at the most detailed level of abstraction available. At the binary level, the display corresponding to the affected flight characteristic is highlighted. At the qualitative level, the status value is displayed by highlighting the corresponding portion of the overlay display described above; and a trend is displayed with an arrow indication in the direction of the changing state.

4.4.2 Effectors

This display (labeled "effector displays" in Figure 4-21) shows the aircraft outline showing the effectors which are mouse-sensitive. It is used during simulation to input faulty effectors and during response derivation to output suggested effector settings.

Input. The user can click on any effector. A menu pops up allowing the selection of binary or qualitative levels. If the binary level is selected, the selected effector's status is set to "abnormal". If qualitative is selected, another menu pops up with the choice of status or trend attributes. When one is selected, a menu with the possible choices pops-up and the user selects the desired value for that attribute.

Output. The suggested effectors are highlighted and, if the reasoning is taking place at the qualitative level, the trend attributes value is overlaid on the effector.

4.4.3 Alarms & Warnings

The alarms and warnings display is used for output only. The alarms and warnings are represented by text strings which are displayed in a separate window. If the alarm or warning has been triggered, the corresponding text string is highlighted. If the reasoning is taking place at the qualitative level, the value is displayed next to the string.

4.4.4 Flight Phase Display

This display shows a graphical representation of the seven flight phases illustrated by an aircraft icon in different positions. The icons are mouse-sensitive.

Input. The user clicks on any of these icons to select the flight phase. This input is used for two purposes:

- To determine for which flight phases simulation or response derivation should be performed.
- To select which results will be displayed on the instrument panel or the effector display, when multiple flight phases have been processed.

Output. The flight phase currently being processed by RECORS is highlighted.

4.4.5 RECORS Control Menus

This black and white display is used both to control RECORS processing and to input and display the aircraft system status. An interim display for inputting flight characteristics is also included.

4.4.6 KEE Development Window Display

This black and white display is used to access the different KEE knowledge bases. The knowledge bases can be viewed in either textual or graphical format. The graphical format shows the taxonomic hierarchy; the textual format shows the individual objects and their slots.

4.5 Future Work

We feel there are several areas in which further research would enhance aiding system effectiveness. This section discusses these directions for future work, which fall into three major categories:

- Basic research in aiding systems in general.
- Basic research in causal modeling.
- Applied work with aiding systems.

For each of these categories we list several possible projects. The section ends with a prioritized list of these projects along two dimensions:

- Importance of work in the area for next phase of the Intelligent Aiding Systems project.
- Level of effort required for the task.

4.5.1 Basic Research in Aiding Systems

This involves addressing the issues of system reliability, verification and validation, the completeness of the representation, and the ability of the system to know when it has reached the boundaries of its expertise. Appendix B discusses verification and validation. The issues associated with real-time response also fall under this category.

4.5.2 Basic Research in Causal Modeling

Work in this area would involve research in knowledge representation and automatic reasoning and would address the following issues:

- limitations of causal reasoning,
- increasing the resolution of the causal interaction to include primary and secondary causal effects,
- representing time and feedback in causal models,
- constructing more detailed models of the aircraft systems,
- combination of taxonomic and causal representations,

- principles of causal model construction, and
- tools for automating the model construction process.

4.5.3 Applied Work in Aiding Systems

Work in this area would involve empirical studies of the interaction between an aiding system and the human crew and would be concerned with the integration of intelligent aiding system and current technology in a number of areas. Some projects falling under this general category are listed below.

- Integration of qualitative reasoning with existing quantitative models (flight director, flight simulators).
- Use of the RECORS prototype for pilot and maintenance crews training.
- Measuring the effectiveness of aided and unaided pilot performance in different situations and for different people with varying levels of expertise. In particular, measuring the effectiveness providing information at the different levels of abstraction.
- Determining at what point on the situation-response spectrum the information should be provided and how does this varies from person to person, by the person's level of expertise, and by the situation.
- How much explanation of the system's reasoning is required and how should it be presented.
- Experiments with different interface designs.

4.5.4 Future Projects Ordered by Importance

1. Representing time and feedback in causal models.
2. Constructing more detailed models of the aircraft systems.
3. Integration of qualitative reasoning with existing quantitative models (flight director, flight simulators).
4. Real-time performance issues.
5. Verification and validation of the aiding system knowledge bases.
6. Measuring the effectiveness of aided and unaided pilot performance in different situations and for different people with varying levels of expertise. In particular, measuring the effectiveness providing information at the different levels of abstraction.

7. Determining at what point on the situation-response spectrum the information should be provided and how does this varies from person to person, by the person's level of expertise, and by the situation.
8. Increasing the resolution of the causal interaction to include primary and secondary causal effects.
9. Determining how much explanation of the system's reasoning is required and how should it be presented.
10. Experiments with different interface designs.
11. Use of the RECORS prototype for pilot and maintenance crews training.
12. Construction of tools for automating the model construction process.

4.5.5 Future Projects Ordered by the Level of Effort Required

1. Representing time and feedback in causal models.
2. Constructing more detailed models of the aircraft systems.
3. Increasing the resolution of the causal interaction to include primary and secondary causal effects.
4. Measuring the effectiveness of aided and unaided pilot performance in different situations and for different people with varying levels of expertise. In particular, measuring the effectiveness providing information at the different levels of abstraction.
5. Determining at what point on the situation-response spectrum the information should be provided and how does this varies from person to person, by the person's level of expertise, and by the situation.
6. Determining how much explanation of the system's reasoning is required and how should it be presented.
7. Experiments with different interface designs.
8. Integration of qualitative reasoning with existing quantitative models (flight director, flight simulators).
9. Real-time performance issues.
10. Verification and validation of the aiding system knowledge bases.

5. References

Abbott, K.H. (1988) Robust Operative Diagnosis as Problem Solving in a Hypotheses in Space *Proceedings of the 7th National Conference on AI*, St. Paul, MN, August, 1988.

Abbott, K.H. (1988) Robust Knowledge-based Fault Diagnosis as Problem Solving in a Hypothesis Space. Draft Ph.D dissertation.

Baron, S. and Feehrer, C. (1985) An Analysis of the Application of AI to the Development of Intelligent Aids for Flight Crew Tasks. BBN Report No. 5937, BBN Laboratories Incorporated, Cambridge, MA.

Bobrow, D.G., and Winograd, T., (1977) An Overview of KRL, a Knowledge Representation Language, *Cognitive Science*, 1, 3-46.

Brachman, R.J., (1977) What's in a Concept: Structural Foundations for Semantic Networks. *International Journal of Man-Machine Studies*, 9, 127-152.

Brachman, R., Ciccarelli, E., Greenfeld, N., and Yonke, M. (1978) KLONE Reference Manual. Technical Report 3848, Bolt Beranek and Newman Inc.

Card, S.K., Moran, T.P., & Newell, A. (1976) The Manuscript Editing Task: a Routine Cognitive Skill. Palo Alto, CA:Xerox Corp., Palo Alto Research Center.

Corkill, D.D., Lesser, V.R., & Hudlicka, E. (1982) Unifying data-directed and goal-directed control: An example and experiments *Proceedings of the Second National Conference on Artificial Intelligence*, pp. 143-147.

Davis, R. (1982) Expert Systems: Where are we and where do we go from here? *AI Magazine*, Vol. 3, No. 2, pp. 3-22.

Draper, W. (1986) Display Managers as the Basis for User-Machine Communication in User Centered System Design. D.A. Norman and S.W. Draper, (Eds.) pp. 339-352.

de Kleer, J. & Brown, J.S. (1985) A Qualitative Physics Based on Confluences, *In Qualitative Reasoning about Physics Systems* D. Bobrow (Ed.) The MIT Press, Cambridge, MA, pp/ 7-83.

Forbus, K.D. (1985) Qualitative Process Theory, *In Qualitative Reasoning about Physical Systems* D. Bobrow (Ed.) The MIT Press, Cambridge, MA pp. 85-168.

Hudlicka, E. & Corker, K. (1988) Integrating Causal Reasoning at Different Levels of Abstraction, *Proceedings of the First National Conference on Industrial and Engineering Applications of AI and Expert Systems*, The University of Tennessee Space Institute, Tullahoma, TN.

Hudlicka, E., Schudy, R., Corker, K, & Baron, S. (1987) Intelligent Aids for Flight Crew Tasks, BBN Report No. 6368, BBN Laboratories Incorporated, Cambridge, MA.

Kuipers, B. (1985) Commonsense Reasoning about Causality: Deriving Behavior from Structure. *In Qualitative reasoning about Physical Systems* D. Bobrow (Ed.) The MIT Press, Cambridge, MA, pp. 169-203.

Mark, W., (1986) Knowledge-Based Interface Design in User Centered System Design D.A. Norman and S.W. Draper, (Eds.) pp. 17-23.

Michie, D. (1981-1982) High-Road and Low-Road Programs *AI Magazine* Vol. 3, No. 1, pp. 21-22.

Minsky, M. (1975) A Framework for Representing Knowledge In P.H. Winston (Ed.), *The Psychology of Computer Vision*, pp. 211-277. New York:McGraw-Hill.

Moran, T.P. (1978) Introduction to the Command Language Grammar. Technical Report SSL-78-3, Xerox Palo Alto Research Center.

Norman, D.A., Draper, S.W. (Eds.) (1986) User Centered System Design. Lawrence Erlbaum Associates.

Norman, D.A., Rumelhart, D.E., et al. (1975) Explorations in Cognition. San Francisco: W.H. Freeman.

National Transportation Safety Board Report No. NTSB-AAR-79-17.

Parasuraman, R. and Davies, D.R. (1984) Varieties of Attention. Academic Press, New York.

Pew, R.W., Woods, W.A., Stevens, A.B., and Weene, P.L. (1978) Identifying Information Requirements for Command and Control Decision Making. Technical Report 3801, Bolt Beranek and Newman Inc.

Proceedings of the 7th National Conference on Artificial Intelligence, St. Paul, MN, pp. 369-374, August, 1988.

Quillian, M.R. (1968) Semantic Memory. In Minsky, M. (Ed.) *Semantic Information Processing*. Cambridge, MA: MIT Press.

Rajagopalan, R. (1984) Qualitative Modeling in the Turbojet Domain, in *Proceedings of the National Conference on Artificial Intelligence*, Austin, TX, pp. 283-287.

Rasmussen, J. (1983) Skills, Rules, Knowledge; Distinctions in Human Performance Models IEEE Transactions of Systems, Man, and Cybernetics SMC-13 (3).

Rasmussen, J. (1986) Information Processing and Human-Machine Interaction An Approach to Cognitive Engineering, North Holland Series in *System Science and Engineering*, Andrew P. Sage (ed.), Series volume 12.

Reisner, P. (1977) Use of Psychological Experimentation as an Aid to Development of a Query Language. IEEE Transactions on Software Engineering, SE-3, 218-229.

Schudy, R.B. (1986) Avionics Expert Systems Definition Study. BBN Report No. 6141, BBN Laboratories Incorporated, Cambridge, MA.

Schneider, B. (1983) Direct Manipulation: a Step Beyond Programming Languages, IEEE Computer, 16(8), pp. 57-69.

Tanner, P. & Buxton, W. (1985) Some Issues in Future Interface Management System (UIMS) Developments. In G. Pfaff (Ed.), *User Interface Management Systems* (pp. 67-69). Berlin: Springer-Verlag.

White, B.Y. & Frederiksen (1987) Causal Model Progressions as a Foundation for Intelligent Learning Environments, *BBN Report 6686* BBN Laboratories Inc., Cambridge, MA.

Williams, B.C. (1984) Qualitative analysis of MOS circuits, *Artificial Intelligence*, Vol. 24, pp. 281-346.

Wiener, E.L., Nagel, D.C., (Eds.) (1986) Human Factors in Aviation Academic Press.

Winston, N. E., Chaffin, R., and Hermann, D. (1986) A Taxonomy of Part-Whole Relations Cognitive Science, Vol. 11, pp. 417-444.

Woods, W. A. (1978) Taxonomic Lattice Structures for Situation Recognition. in TINLAP-2, *Conference on Theoretical Issues in Natural Language Processing-2*, University of Illinois at Urbana-Champaign.

Woods, W.A. (1975) What's in a Link: Foundations for Semantic Networks. In *Representation and Understanding* Bobrow, P.G. and Collins, A. (Eds.) New York: Academic Press.

Woods, W.A. (1970) Transition Network Grammars for Natural Language Analysis. *Communications of the ACM*, 13, 591-606.

Woods, W.A. (1986) Cognitive Technologies: The Design of Joint Human-Machine Cognitive Systems, *AI Magazine*, Vol. 6, No. 4, pp.86-92.

Appendix A
Aircraft Incidents/Accidents

Incident 1: An illustration of the value of onboard expert assistance is provided by an incident on a Miami to Nassau flight, May 1983. In that flight an L-1011 received low oil warnings in all three engines. The flight engineer realizing the extremely low probability of all three engines failing simultaneously decided to eliminate other causes for the readings. (The pilot in the meantime was taking precautionary, but not full emergency, measures.) In order to check the possibility of sensor fault or a common electrical basis for the readings, the flight engineer needed to contact ground technicians for detailed information on sensor operation and electrical bus connections. Transmission of symptoms to ground and instructions to crew for test procedure was an involved and lengthy process. It was determined that the sensors were accurate and full emergency landing procedures were implemented. A disaster was narrowly avoided. An on-board diagnostic expert may have offered more timely advice on the nature of the emergency.

Incident 2: This is a well studied case of maintenance procedure, flight procedure, and situation assessment failures. In that accident, an engine separated from the left wing approximately the time of aircraft rotation and lift off. In separating, the engine tore off the leading edge slats, which increased the minimum flight speeds necessary to prevent stall. The damage rendered primary and secondary slat controls, slat disagreement, and stall warning systems inoperative. The flight crew reduced aircraft speed and climb angle as per the standard company procedures for climbout with a failed engine. The loss of slat disagreement and stall warning indicators prevented the crew from realizing that by following the prescribed procedure they were inducing asymmetrical stall of the left wing, which resulted in a roll which was uncontrollable at the low flight speed.

Incident 3: On a Boeing 737 flight from Philadelphia to Rochester in July of 1970 an explosion was heard as the aircraft cleared 50 feet altitude at takeoff. The captain joined the first officer on the controls and decided to abort take-off because he perceived no response to advanced power to both engines. The captain thought that the right engine had failed while the

first officer correctly perceived that the left engine had failed. There was a problem in control transition because both pilots were flying and it was difficult for the captain to distinguish between first officer commanded inputs and engine failure effects. The result was an aircraft accident due to an inability to disambiguate a single from dual engine failure, and to distinguish the effects of flight crew actions from the effects of engine failure. An intelligent assistant system could probably have prevented this accident by providing the captain with the key attribute which he required to assess the situation: the status of both engines (failed or normal).

Appendix B
Information Processing Model Verification/Validation

In order to validate a model of human information processing in a complex interactive system, the source of the complexity must be established and an assessment of the impact of that complexity must be provided. Complex interactions in man/machine system arise in the interaction of the task's demands, the machine's demands and capabilities, and the human's abilities to respond to demands and to exploit system capabilities. If the demands of the system are low, then an interface system imposes no serious tax on the operator's capability. Correspondingly, if the capabilities of the operator are formidable then system requirements are likely to be met. The shape of this intersecting space is not regular. There are, within a given pilot/aircraft situation, asymmetric capabilities and loadings. The system designer is attempting to achieve a balance between demands and capabilities. The researcher is attempting to provide him the basis for achieving that balance. The human information processing model that underlies the pilot interface management system architecture requires (and supports) empirical investigation to validate its component assumptions.

Complex cognitive behavior is represented in our model as a goal-directed, multi-staged, adaptation to environmental demands. That adaptation has two components: The innate information processing characteristics of the human operator, and learned/skilled responses to the situation's demands.

In the context of our system validation these components translate into two foci in empirical research.

First, there is research required to validate the tiered and staged information processing structure. This research concern encompasses such diverse psychological considerations as perceptual signal-to-noise ratios, decision criterion shifts, attention, active memory constraints, organization of long-term memory and recall, logical demands, response selection, and such parameters of response execution as time, effort, topology, complexity, and display/response compatibility.

Tests of the basic appropriateness of the human information processing model stem from the predictions of that model. The following hypotheses provide a non-exhaustive list of the types of issues engendered by this approach. The time required to unambiguously define a situation should follow an increasing function of the number of attributes related to that situation. Movement among and between levels of situation hierarchies should involve some measurable workload. Mental effort in mapping situations to responses should be dependent on the level to which response is defined and the requirement for conflict resolution. Temporal loading or other resource constraints should affect the number of attributes considered in situation identification and effect the level in the abstraction hierarchy at which the pilot is most comfortably operating. There are, in fact, a host of cognitive-structural issues to be investigated in order to fully substantiate the underlying processing model.

Second, there is a requirement to objectively assess the appropriateness of the the pilot/system interface design. Specifically, there is a need to develop techniques for representation of "situations" (as opposed to system attributes) in pilot displays. Within the representation of situations there is concern for:

- Appropriate displays of uncertainty: This uncertainty can occur at several levels of the situation/response architecture, and should therefore be tailored to the locus of confusion.
- The nature of explanation: how to explain to the pilot where or how the situation was identified (in the context of engine failure), and where and how response aiding was selected.
- The case of response aiding: there are issues as to how to provide advice both for remedial action on the part of the crew, or advice as to how to further help the intelligent assistant in making diagnosis, i.e., advise what data to gather to distinguish or disambiguate a situation assessment.
- Finally there are issues as to how and when to provide "state" information to the pilot.

Subsuming all of these issues are questions as to the appropriate display format and modality.

In response to these issues, a sequence of experimental validation tests could be performed for potential display design configurations. These tests would attempt to exercise the situation/response model by establishing engine failure conditions of several types across several

situations. Subjects would be required to identify failure conditions and suggest remedial action. The appropriateness of response could be determined by reference to air transport policy and expert pilot opinion.

The third area of concern in model validation and interface design has a more theoretical and dynamic basis. It concerns the movement among levels of representation in a knowledge structure. If the situation description is structured hierarchically and the pilot is interrogating the system state at a fairly low level, how does s/he move in response to a perceived requirement to change representation levels or in response to changes in situation. The issues deal with how to tailor a display to the particular diagnostic level being applied by the flight crew, and how to keep them informed as to the level it at which they are is operating.

NASA

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-181905		

4. Title and Subtitle	5. Report Date
Flight Crew Aiding for Recovery from Subsystem Failures	December 1989
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
E. Hudlicka, K. Corker, R. Schudy, and S. Baron	Report No. 6990
	10. Work Unit No.

9. Performing Organization Name and Address	11. Contract or Grant No.
BBN Systems and Technologies Corporation 10 Moulton Street Cambridge, Massachusetts 02138	NAS1-17335
	13. Type of Report and Period Covered
	Contractor Report

12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Langley Research Center Hampton, Virginia 23665-5225	

15. Supplementary Notes

Langley Technical Monitor: Kathy Abbott
Final Report

16. Abstract

This document discusses some of the conceptual issues associated with pilot aiding systems and describes an implementation of one component of such an aiding system: a recovery recommendation system (RECORS).

The first part of this document focuses on aiding systems design issues. Since an aiding system, by definition, acts to assist the crew, it is important that the format and content of the information it presents be compatible with the crew's mental models of the task. We have selected Rasmussen's general information processing strategy to serve as the bridge between the human and aiding system's information processes.

The second part of this document describes the development and implementation of a model-based situation assessment and response generation system for commercial transport aircraft emergencies. This system (RECORS) uses a causal model of a subset of the flight domain to simulate the effects of failures and to generate responses. Since detailed information about the aircraft state may not be available, the model represents the domain at varying levels of abstraction and uses the less detailed abstraction levels to make inferences when exact information is not available.

RECORS is intended to be integrated with the FaultFinder system currently under development at NASA-Langley.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
	Unclassified - Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	94	

NASA FORM 1626 OCT 86

End Date May 22, 1990